Titre: Title:	Detecting Specific Types of DDoS Attacks in Cloud Environment by Using Anomaly Detection
Auteur: Author:	Hossein Abbasi
Date:	2015
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Abbasi, H. (2015). Detecting Specific Types of DDoS Attacks in Cloud Environment by Using Anomaly Detection [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1916/

UNIVERSITÉ DE MONTRÉAL

DETECTING SPECIFIC TYPES OF DDOS ATTACKS IN CLOUD ENVIRONMENT BY USING ANOMALY DETECTION

HOSSEIN ABBASI

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE INFORMATIQUE)

AOÛT 2015

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

DETECTING SPECIFIC TYPES OF DDOS ATTACKS IN CLOUD ENVIRONMENT BY USING ANOMALY DETECTION

présenté par: ABBASI Hossein

en vue de l'obtention du diplôme de: Maîtrise ès Sciences Appliquées

a été dûment accepté par le jury d'examen constitué de:

M. PIERRE Samuel, Ph. D., président

Mme BELLAÏCHE Martine, Ph. D., membre et directrice de recherche

M. TALHI Chamseddine, Ph. D., membre et codirecteur de recherche

M. BEAUBRUN Ronald, Ph. D., membre

DEDICATION

To:

My love:

Roja

My parents:

Ehsan & Mahrokh

My sister and brother:

Samaneh & Reza

Their constant support and encouragement made this possible

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere gratitude to my supervisors, Professor Martine Bellaiche and Professor Chamsedine Talhi, for their extensive support throughout the program.

I would like to convey my gratitude to Professor Samuel Pierre and Professor Ronald Beaubrun for accepting to be a jury member.

A sincere appreciation goes to Professor Michel Dagenais and his laboratory (Dorsal) for their support in certain parts of this project.

I would like to sincerely thank my colleagues and friends during this work. Specifically, I must acknowledge Dr. Babak Khosravifar for their great help in revising this thesis.

Finally, I would like to convey my heartfelt thanks and special gratitude to my family: to my parents who provided unconditional supports throughout my educations and devoted all their life to my success, to my beloved, Roja, for her unwavering love, patience and emotional support; to my sister, Dr. Samaneh Abbasi, who is a gift from God to me to support me in all of my endeavors, and last but not the least to my brother, Reza, for his support and love. I wish you health, happiness and just everything your heart desires.

RÉSUMÉ

Un des avantages les plus importants de l'utilisation du cloud computing est d'avoir des services sur demande, et donc la méthode de paiement dans l'environnement du cloud est de type payer selon l'utilisation (pay per use). Cette caractéristique introduit un nouveau type d'attaque de déni des services appelée déni économique de la durabilité (Economic Denial of Sustainability EDoS) où le client paie des montants supplémentaires au fournisseur du cloud à cause de l'attaque. Les attaques DDoS avec leur nouvelle version sont divisées en trois catégories:

1) Les attaques de consommation de la bande passante.

2) Les attaques qui ciblent des applications spécifiques.

3) Les attaques d'épuisement sur la couche des connections.

Dans ce travail, nous avons proposé un nouveau modèle pour détecter précisément les différents types des attaques DDoS et EDoS en comparant le trafic et l'utilisation des ressources dans des situations normale et d'attaque. Des caractéristiques (features) qui sont liées au trafic et à l'utilisation des ressources dans le cas de chaque attaque ont été recueillies. Elles constituent les métriques de notre modèle de détection. Dans la conception de notre modèle, nous avons utilisé les caractéristiques liées à tous les 3 types d'attaques puisque les caractéristiques d'un type d'attaque jouent un rôle important pour détecter un autre type. En effet, pour trouver un point de changement dans l'utilisation des ressources et le comportement du trafic nous avons utilisé l'algorithme des sommes cumulées CUSUM. La précision de notre algorithme a ensuite été étudié en comparant sa performance avec celle d'un travail populaire précédent. Le taux de détection du modele était élevé, Ce qui indique la haute précision de l'algorithme conçu .

ABSTRACT

One of the most important benefits of using cloud computing is to have on-demand services; accordingly the method of payment in cloud environment is pay per use. This feature results in a new kind of DDOS attack called Economic Denial of Sustainability (EDoS) in which the customer pays extra to the cloud provider because of the attack. DDoS attacks and a new version of these attacks which called EDoS attack are divided into three different categories: 1) Bandwidth–consuming attacks, 2) Attacks which target specific applications and 3) Connection–layer exhaustion attacks. In this work we proposed a novel and inclusive model to precisely detect different types of DDoS and EDoS attacks by comparing the traffic and resource usage in normal and attack situations. Features which are related to traffic and resource usage in each attack were collected as the metrics of our detection model. In designing our model, we used the metrics related to all 3 types of attacks since features of one kind of attack play an important role to detect another type. Moreover, to find a change point in resource usage and traffic behavior we used CUSUM algorithm. The accuracy of our algorithm was then investigated by comparing its performance with one of the popular previous works. Achieving a higher rate of correct detection in our model proved the high accuracy of the designed algorithm.

TABLE OF CONTENTS

DEDICATION III

ACKNOWLEDGEMENTS IV

RÉSUMÉ V

ABSTRACT VI

TABLE OF CONTENTS VII

LIST OF FIGURES XI

LIST OF TABLE XIII

LIST OF SYMBOLS AND ABBREVIATIONS XIV

CHAPTER 1 INTRODUCTION 1

1.1 Motivation 3

1.2 Research Objectives 4

1.3 Thesis outline 4

CHAPTER 2 LITERATURE REVIEW AND BACKGROUND 5

2.1 Cloud computing 5

2.2 Virtualization 10

2.2.1 Full virtualization 11

2.2.2 Para virtualization 11

2.2.3 Application virtualization 12

2.2.4 Hardware support virtualization 12

2.2.5 Resource virtualization 13

2.2.6 Container Virtualization 13

2.3 Cloud standards 13

2.4 Cloud security 14
2.4.1 Availability 14
2.4.2 Confidentiality 15
2.4.3 Data Integrity 16
2.4.4 Control 16
2.4.5 Audit 16
2.5 Security challenges 17
2.5.1 Rootkit Attacks 17
2.5.2 VM Escape 17
2.5.3 VM monitoring from another VM 18
2.5.4 Guest-to-Guest attack 18
2.5.5 Denial of service attack 18
2.5.6 Economic Denial of Sustainability (EDoS) (Special DDOS attack for cloud computing) 23
2.6 Related work in Defence and Detection algorithm against DDOS and DOS 23
2.6.1 Defense and Detection Solutions for Cloud Environments 23
2.6.2 Defense and Detection Solutions for non-Cloud Environments 26
2.7 Related work in detection and defense against EDoS attacks in cloud computing 27
2.8 Open problem 29
CHAPTER 3 METHODOLOGY 32
3.1 Extracting the information for the normal traffic and resources 32
3.2 Monitoring and Sampling Module 33
3.3 Attack identification 34
3.4 Metrics 35
3.4.1 Time Spent on a Web Page (TSP) 36

3.4.2 Network I/O in webserver virtual machine and Database virtual machine.....37
3.4.3 Percent of CPU usage in webserver virtual machine and database virtual machine..37
3.4.4 Memory usage in webserver virtual machine 38
3.4.5 Network bandwidth in webserver 38
3.4.6 Packet information 39
3.4.7 Number of packets per second (incoming and outgoing) 39
3.4.8 Number of half-opened connections 40
3.5 Change Point Detection algorithm: CUSUM.....40
3.6 Proposed algorithm to detect different types of DDoS and EDoS attacks.....43
3.6.1 Http attack sings.....43
3.6.2 Database attack signs 44
3.6.3 TCP SYN Flood attack signs 44
3.6.4 Detecting changes by using CUSUM algorithm:.....48
3.6.5 Detecting attack percentage procedure 51
3.7 Conclusion.....52
CHAPTER 4 RESULTS AND DISCUTIONS 53
4.1 Set up for experimental result 53
4.2 Time Spent on a Web Page (TSP).....55
4.3 Network I/O in webserver virtual machine and Database virtual machine.....56
4.4 Percent CPU usage in webserver virtual machine and database virtual machine.....58
4.5 Memory usage in webserver virtual machine 60
4.6 Network bandwidth in webserver 61
4.7 Packet information 65
4.8 Number of packets per second (incoming and outgoing) 67

4.9 Number of half-opened connections 68
4.10 Evaluation and discussion 69
4.10.1 Accuracy evaluation 70
4.10.2 Metrics evaluation 71
CHAPTER 5 CONCLUSION 74
REFERENCES 76

LIST OF FIGURES

Figure 2.1: The overall structure of the cloud as defined by NIST[11] 7

Figure 2.2: Cloud architecture 9

Figure 2.3: Full virtualization 11

Figure 2.4 : Para virtualization(Xen) 12

Figure 2.5: DoS attack (above) and DDoS attack (middle) in old system and DDoS attack in cloud computing system (below) 20

Figure 3.1: Detecting and monitoring system 34

Figure 4.1: Variation of TSP values (left) and CUSUM values in TSP(right) in Http attack 55

Figure 4.2: Variation of TSP values (left) and CUSUM values in TSP(right) in Database attack 56

Figure 4.3 : I/O usage in webserver virtual machine in normal situation by using IPTraf 57

Figure 4.4:Different amount of network I/O in different situation 58

Figure 4.5 : CPU usage 59

Figure 4.6: CPU usage in different situation 60

Figure 4.7: Variation of MemW values (left) and CUSUM values in MemW (right) in TCP SYN Flood attack 61

Figure 4.8 : Bandwidth usage per hour 62

Figure 4.9 : Bandwidth usage per day for an 8-day period (from 06/26/2014 to 07/03/2014) 62

Figure 4.10 : Bandwidth usage per week 63

Figure 4.11 : Bandwidth usage per month 63

Figure 4.12:Variation of NBWph values (left) and CUSUM values in NBWph (right) in Http attack 64

Figure 4.13:Variation of NBWpd values (left) and CUSUM values in NBWpd (right) in Http attack 64

Figure 4.14:Variation of NBWpw values (left) and CUSUM values in NBWpw(right) in Http attack 65

Figure 4.15: Variation of NBWpm values(left) and CUSUM values in NBWpm(right) in Http attack 65

Figure 4.16 : Packets information using LTTng 66

Figure 4.17 : TCP Header 67

Figure 4.18:Variation of NPi values (left) and CUSUM values in NPi(right) in TCP SYN Flood attack 67

Figure 4.19:Variation of NPo values (left) and CUSUM values in NPo(right) in TCP SYN Flood attack 68

Figure 4.20 : Number of half-opened connection by use of netstat 68

Figure 4.21: Variation of NHOP values(left) and CUSUM values in NHOP(right) in TCP SYN Flood attack. 69

Figure 4.22: Rate of correct detection comparison 70

Figure 4.23: Testing our framework metrics by Neural Network 72

Figure 4.24 : Evaluating our metrics by using Neural network 73

LIST OF TABLE

Table 3.1 : Relationship between R(SYN), R(ACK) and R(SYN+ACK) 46

Table 4.1 : System details 54

LIST OF SYMBOLS AND ABBREVIATIONS

CPUD	Percent of CPU usage in Database virtual machine.
CPUW	Percent of CPU usage in Webserver virtual machine.
CUSUM	Cumulative sum
DDoS	Distributed denial-of-service
DoS	Denial-of-service
EDos	Economic Denial of Sustainability
IDS	Intrusion detection system
IODi	Network I/O in Database virtual machine (incoming) (Kbits/s).
IODo	Network I/O in Database virtual machine (outgoing) (Kbits/s).
IOWi	Network I/O in Webserver virtual machine (incoming) (Kbits/s).
IOWo	Network I/O in Webserver virtual machine (outgoing) (Kbits/s).
MemW	Percent of Memory usage in Webserver virtual machine.
NBWph	Network bandwidth usage in Webserver per hours (MB).
NBWpd	Network bandwidth usage in Webserver per day (MB).
NBWpw	Network bandwidth usage in Webserver per week (GB).
NBWpm	Network bandwidth usage in Webserver per month (GB).
NHOP	Number of half opened connection.
NPi	Number of packet (incoming) per second.
NPo	Number of packet (outgoing) per second.
R(SYN)	The ratio of SYN packets in TCP packets.
R(ACK)	The ratio of ACK packets in TCP packets.
R(SYN+ACK)	The ratio of SYN and ACK packets in TCP packets.
TSP	Time Spent on a Web Page.

CHAPTER 1 INTRODUCTION

Cloud computing is a revolutionary concept that has transformed the information and communication technology by delivering computational resources as services across the internet. Cloud computing provides inexpensive and scalable resources on demand to system requirement and consequently, there is no need to invest in a huge computer system.

Users are not the owner of computing server. They can access to a numerous services without the infrastructure cloud management. Application and data are distributed in the cloud. Most of the time, the well connected servers allow the user to access the data only through a web browser.

However, security is a big concern in this new technology. There are more systems to protect, more possible points of entry, more holes to patch and also more interconnection points in the cloud and accordingly the security in the cloud is more critical than old system [1]

Denial of service (DOS) attack is a popular problem in network security. DOS attack increases the server load and makes the system out of reach[2]. The absolute prevention of the occurrence of Distributed Denial of service (DDoS) attacks, which are the most popular types of DoS attacks, is not possible; therefore, detecting these attacks is an important step in securing the server against this kind of very common security threat.

According to the general definition of the National Institute of Standards and Technology (NIST), providing On-demand services is the most important benefit of using cloud computing [3]. It means that cloud customers don't need to buy whole resources and infrastructure for the first time. The method of payment in cloud environment is pay per use[4] . Based on this benefit of cloud computing, There is a specific type of DDoS attack specially designed for cloud computing environment which is called Economic Denial of Sustainability (EDoS)[5].

Every type of DoS attacks (or Distributed type of DoS) that happened in cloud computing and caused an economic problem can define as EDoS attack as well.

In fact, there are two ways for implementing this kind of attack in cloud environment. In the first one, an attacker who is outside of the cloud performs and sends (D)DoS to a VM; while in the second one an attacker from inside of the cloud implements D(D)oS which means attacker uses a VM to send packets and attack to hypervisor or another VMs. But it is essential to note that in both types, attacker try to use of sharing resources to make all cloud become out of service it

means effect of this attack finally influence to hypervisor and eventually will be harm for all cloud. At the first glance it may look like that the attack is similar to a traditional attack (attack in non-virtualization environment) when we hypothesize that the attacker is outside of the cloud and victim is a server of a company that is in a VM; but in fact when a server is in VM, resources of this server are extendable and increasable. According to this fact every DDoS attacks in this server can change to EDoS attack that is a cloud special attack.

Also on demand resources which is an important benefit of cloud computing, can be a big concern for the cloud because it can transfer an attack from VM to hypervisor. When a (D)Dos attack happens in VM, the resources of this virtual machine will be ended in a little time so VM wants to increase its resources to avoid being out of service; thus VM applies for more resources and sends this request to the hypervisor; by doing so, the attack is transmitted to the hypervisor, after a few minutes and accordingly using the live migration technique the entire cloud system will be crashed and become out of service.

In this work we consider that virtual servers in cloud computing can be targeted by three general classes of attacks:

1) Bandwidth– consuming attacks: This kind of attacks uses the total bandwidth of the target with large volumes of data packets. It leads to denial of services because normal requests cannot receive quick and operational responses [6]. In this work we use HTTP attack as a good representative for the Bandwidth – Consuming EDoS.
2) Attacks that target specific applications: This kind of attacks targets a special application in servers. In cloud computing environment each application can run in a virtual machine; therefore, in an attack scenario, the attacker tries to harm servers by attacking the special VM that includes special application. In our work we chose Database as the application which is targeted by this type of attacks (Attacks that target specific applications). The virtual machine allocated to Database receive many database queries for each HTTP request in case of Database attacks [7, 8].
3) Connection–layer exhaustion attacks: This kind of attacks tries to use the protocol features like Three-Way Handshake in TCP/IP protocol to attack the servers. TCP SYN Flood, ICMP/ UDP flood attacks are examples of connection–layer exhaustion attacks. Since these attacks happen at the time of making connection between client and server, we called this category" connection-layer exhaustion attacks". As a delegate for

connection–layer exhaustion attacks at this work, we nominate TCP SYN Flood attack which is one of the most important issues in security aspect[2];

Our main contributionin in this work is to detect different types of DDoS and EDoS attacks in a framework which has not been considered in previous work.

1.1 Motivation

Since cloud environment is elastic and works based on the pay as you use model, additional resources are easily available; but the customer has to pay extra for them. Economic denial of sustainability is a scenario in which the customer pays extra to the cloud provider because of the attack. Also, because in this situation DDoS attack is transferred to the hypervisor, it can be dangerous for the hypervisor and finally for the entire cloud.

The first prevention method that comes to mind is to limit the resource allocation[9]; however, by doing so we actually limit some of the most important advantages of the cloud computing . Thus, the absolute prevention of the occurrence of Distributed Denial of Service (DDoS) attacks and Economic Denial of Sustainability (EDoS) attacks in cloud computing is not possible; therefore, detecting these attacks is an important step in securing the cloud against this kind of very common security threat.

The majority of existing algorithms for detection DDoS attacks in cloud computing environment work based on packet information; however, the packet in DDoS attack is like packet in normal situation and the only difference is that there are too many packets in DDoS attack; consequently the current packet based approaches are not reliable and powerful enough for DDoS detection.

On the other hand, the present algorithms for EDoS attacks have mainly focused only on finding a solution for the defense and mitigation of EDoS attacks. Moreover, they are useful only for one attack and to our best knowledge there is not any global model to detect all types of EDOS.

Therefore we decided to detect DDoS and EDoS attacks in cloud computing by working on traffic and resource usage anomaly detection.

Our main idea for detecting these attacks is that, even if attackers in DDoS and EDoS attacks can make packets like normal packets but the traffic generated in DDOS attack isn't like traffic in normal situation. Also the proportion of resource usage is completly different in case of attack in

compare with normal situation. So we want to get some sample of traffic and resource usage in cloud computing servers and work on them to find a good approach to detect these 3 types of attacks (HTTP attack, Database attack and TCP SYN Flood attack).

Thus our contributions in this work are as follow:

1- Detecting DDoS and EDoS attacks by working on traffic and resource usages anomaly detection in cloud environment.

2- Introducing an inclusive algorithm which detects HTTP attack, Database attack and TCP SYN Flood attack no matter what time each attack happens.

In chapter 3, we will explain our methodology in more detail.

1.2 Research Objectives

Our main goal in this work is to design and develop a model to detect different types of DDoS and EDoS attacks in the cloud computing. Our specific objectives are as follow:

1. To extract information for the normal situation (traffic and resource).
2. To propose a module for sampling the traffic and resource
3. To identify the attack
4. To define the relevant metric for each attack
5. To propose an algorithm for an abrupt change for the relevant metric
6. To develop detection method for different types of DDoS and EDoS attacks.

1.3 Thesis outline

This thesis is organized in several chapters, the first chapter being this “Introduction. Chapter (2) presents the literature review and background, chapter (3) provides our methodology for Detecting Different Types of DDoS and EDoS attacks, chapter (4) provides results and discutions and finally chapter (5) discusses about Conclusion and future work.

CHAPTER 2 LITERATURE REVIEW AND BACKGROUND

In this section we study the state of art research in cloud computing, its aspects, benefits and problems. Section (2.1) discusses about cloud computing, section (2.2) discusses type of virtualization that is one of the main concepts in cloud computing, section (2.3) provides some cloud standards, section (2.4) is about cloud security, section (2.5) discusses challenges in cloud security and some solutions for them, section (2.6) is a related work in detection and defense against DDOS and DOS attacks,we discuss related work in detection and defense against Economic Denial of Sustainability (EDoS) attacks in cloud computing in section (2.7) and finally section (2.8) introduces open problem.

2.1 Cloud computing

The concept of cloud computing was introduced for the first time in 1961 by John McCarthy[10] who predicted that the computers may eventually be organized in a way to be used as a general tool. After that, the telecom companies offered virtual private networks that significantly reduced the cost of service provision[4]. In 2007, large companies such as Amazon, Google, IBM and prestigious universities in the world began research projects in the field of cloud computing. In fact, the positive and striking features of cloud computing caused many large and reputable organizations to be turning to it.

In this section we collect some definitions of cloud computing that each of them want to explain cloud computing.

Buyya et al.,[11] have defined the cloud as a type of parallel and distributed system that consists of a set of virtual computers that as a single unit or multiple computational resources are presented. This definition is based on service level negotiated between service providers and consumers.

Cohen believes that cloud computing is one of the achievements is trying to make progress in various aspects (load balancing - business models - models, architecture, etc.), as well as a new step in providing cloud computing software. For them, the simplest definition for cloud computing is a middle software for internet [1].

Kaplan believes that cloud computing is like the array of services in the web and its target is that applications achieve to large scale of the ability to make choices based on the “Pay-as-you-go “.

Cloud computing is an understanding of Utility computing without the technical complexities or concerns about the complexity extension on it [1].

According to Sheynkman cloud computing focuses to make a layer over the hardware that can be used upon request. Many companies require hardware virtualization environment can be easily configured, deployed, to be managed dynamically expanded for take complete control of clouds and infrastructure applications [1].

As it is clear in above paragraphs, a lot of definitions for cloud computing is presented by the authors, and there is still a debate and an uncertainty over the exact definition of cloud computing in industry. In January 2011 the National Institute of Standards and Technology (NIST) offered a general definition "Cloud computing is a model that enables the on-demand network access to configurable computing resources at each site, appropriately" [3]. Networks, servers, storage sites, applications, and services can be examples of these resources.

Today almost all enterprises consider the Cloud computing as a trend scenario and they try to make an entry in it. The benefits of utilizing cloud computing are[4]:

I) Reduce the cost of equipment and maintenance.

II) Improve accessibility in worldwide.

III) Provide a flexible and highly automated process where there is no concern about software up-gradation which tends to be a matter of every day.

IV) Payments based on scalability.

V) Using internet technology.

VI) Self-service based on demand of high performance.

VII) Rapid implementation, ease of maintenance and updates.

There are some drawbacks in using cloud computing as well. Recovering the data, lack of control over cloud services, service level agreements, legal problems, existing of different architectures, auditing, reviewing and evaluating the performance of cloud computing environment, are the major flaws in cloud computing[4].

The model that is provided by NIST is combining five essential characteristics, three service models and four models of deployment that are depicted in Figure 2.1.

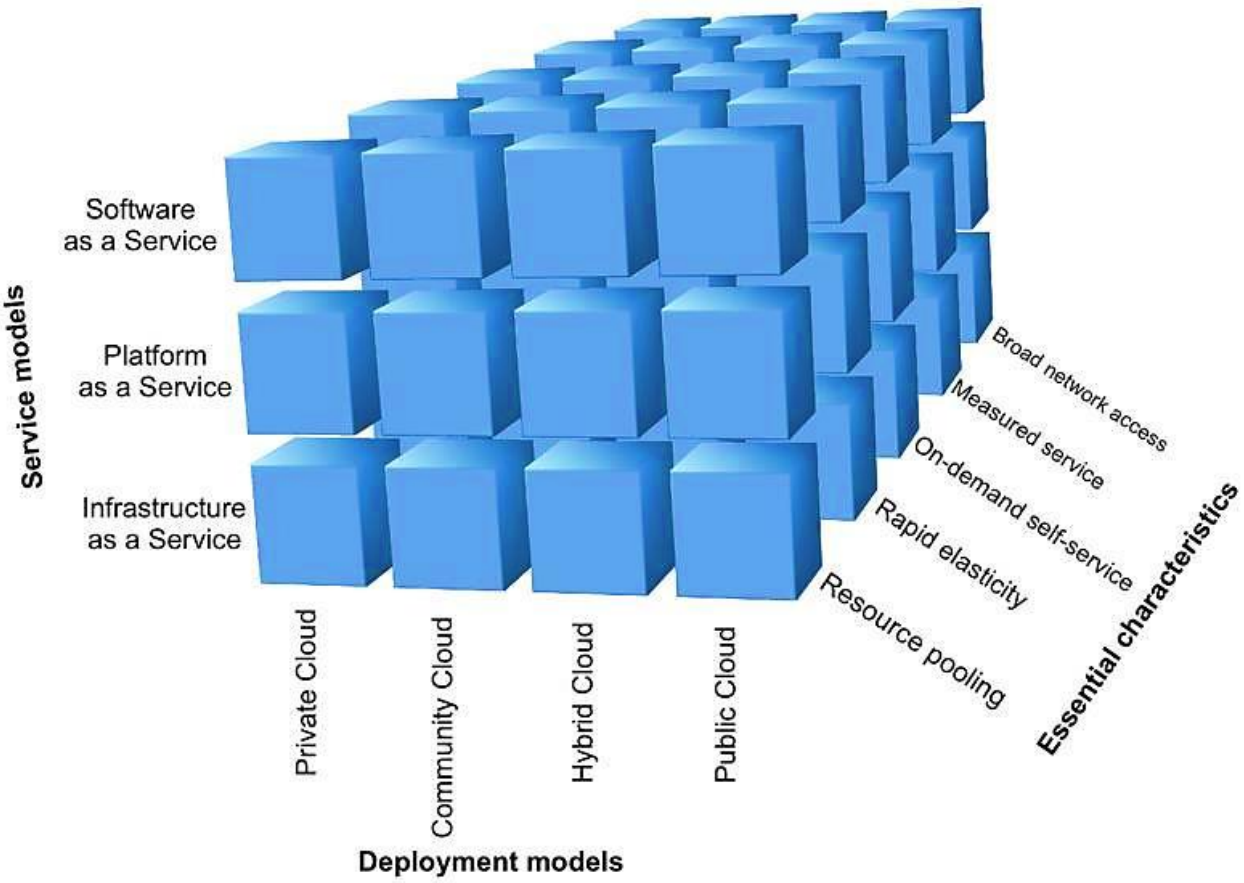


Figure 2.1:The overall structure of the cloud as defined by NIST[12]

There are four main models of cloud computing deployment in NIST definition; cloud services that we are going to mention them in next step can be deployed in four ways depending customer requirements, they also are mentioned Cloud computing variety in some of sources. Four main models of cloud computing deployment are as follow[4]:

1) **Public Cloud:** in this model, a cloud infrastructure which is provided to many customers is organized by a third party[13]. Multiple enterprises are able to perform their tasks on the infrastructure simultaneously. Users dynamically supply their resources via the web from a service provider offsite. A waste of resources is chosen by the users and they pay based on their usage.

2) **Private Cloud:** a cloud infrastructure which is available only for a specific client and organized by the company itself or a third party service provider[13]. It is based on the concept of virtualized machines, and is an exclusive network.

3) **Community cloud:** an Infrastructure which is shared by several enterprises for a common reason and can be organized by them or a third party service provider[14].

4) **Hybrid Cloud:** the composition of two or more cloud distribution patterns, linked in a way to put together the transferred data without interfering with each other[15].

At a glance, a cloud must have the following five characteristics to be considered as a cloud[3]:

- On demand self-service: cloud computing should work in base on "Pay-as-you-go".
- Accessing the WAN: Permanent access to internet is one of the main requirements for have a cloud computing. Since in some country some people don't have permanent access to high speed internet, it is possible that they have problem for migrate their services in to cloud.
- Resource sharing (Sharing of pooled resource): split a physical computer into several machines for economize especially in cost.
- Potential flexibility (Rapid elasticity): cloud should provide a flexible and highly automated process where there is no concern about software up-gradation which tends to be a matter of every day. Also cloud computing should have rapid implementation, ease of maintenance and updates.
- A set of services (Metered Services): A cloud should have a set of service to provide to customer, without services a cloud is meaningless.

Based upon the types of the services, three categories may be included in the cloud computing [12, 16] that you can see in Figure 2.2:

1) **Infrastructure as a Service (IaaS):** The lowest level that provides basic support infrastructure service. In order to control the creation and utilizing of virtual machines a governance framework is required also in this category the unauthorized access to sensitive information of the user is prevented. For example in Amazon Elastic Compute

Cloud (Amazon EC2) provide VMs to their customer for using of resources. Amazon provides its customer with complete control of customer's computing resources[17].

2) **Platform as a Service (PaaS):** It is the intermediate layer and in addition to providing environment to host user applications it provides services focused on the platform. Microsoft azure is a platform for cloud computing systems. It is like an OS that allow to customer to quickly build, deploy and manage applications[18].
3) **Software as a Service (SaaS):** It is the top layer that has a complete application offered as a service on demand[19]. SaaS guarantees that the entire applications are hosted on the web where the users utilize them. Payment is based on pay-per-use. In this category, there is no need to install and run application on the local computer of the client, thus the customer's responsibility for the maintenance of the software is remarkably reduced. Google App, Facebook and Youtube are some example of SaaS that end users are connects with them in cloud.

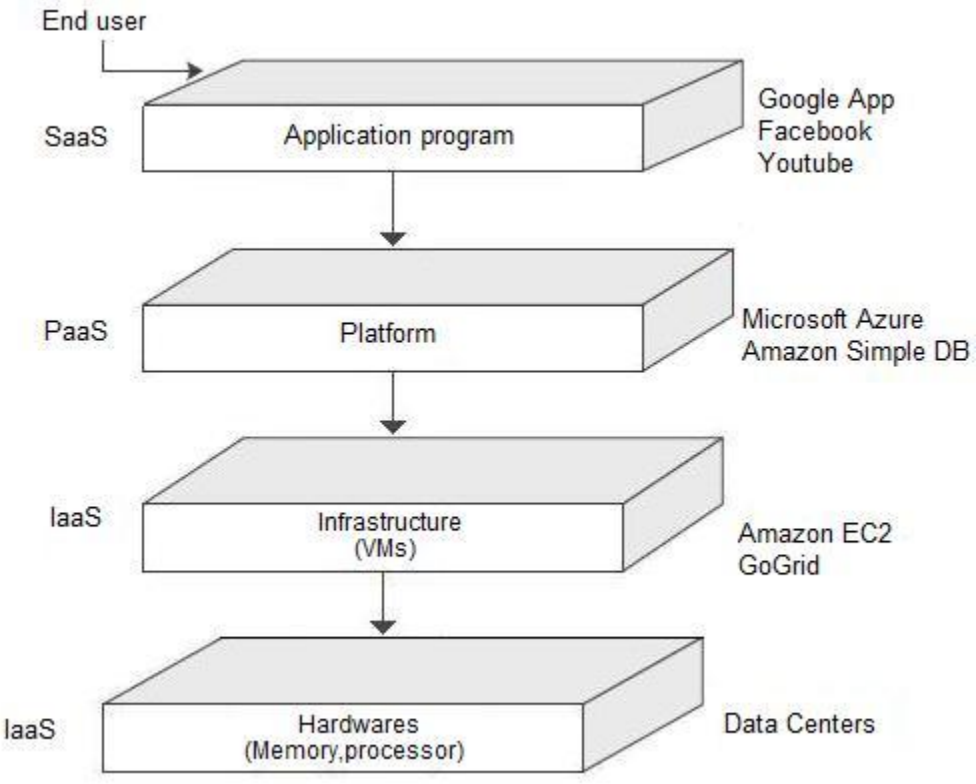


Figure 2.2:Cloud architecture[4]

2.2 Virtualization

Virtualization is the main concept in the cloud computing. Virtualization is a method that splits a physical computer into several machines which are partly or completely isolated and generally known as virtual machines (VM) or guest machine [20]. Recently a lot of attention has been paid to the concept of virtualization as a crucial technology in terms of both academic and industrial applications. An outstanding example is the Xen virtualization systems that have been used by current market leader in the Amazon EC2 cloud computing to provide customers with computing resources. Using Virtualization the organizations can remarkably reduce the operational cost while still ensuring enhanced efficiency, boosted utilization and better flexibility of existing hardware. A Survey shows that virtual machines are being used in 90% of organizations in some capacity in their IT infrastructures[21]. A further 34% utilize the virtualization for the majority of their server requirements. Virtualization has been a fundamental concept in the recent years due to a huge rise of cloud computing in the modern era. The main issue is providing an environment that has the capability of rendering all the services to the end users, by support of a hardware which is observed on a personal computer.

Virtualization is a concept that greatly attributes to the security implications in the environment. One of the most important components in Virtual Machines (VMs) is the hypervisor which is a piece of computer software, firmware or hardware that creates and runs virtual machines. In other words, hypervisor is a software layer that locates between the VMs and the physical hardware to implement the Virtualized environments, and to allocate resources to the VMs, such as main memory and peripherals [22]. The hypervisor or Virtual Machine Monitor (VMM) can also act as a gatekeeper to the underlying hardware to improve performance isolation between guests on a host.

The security requirements are much higher in virtualization due to the existence of more systems to protect, more possible points of entry, more holes to patch and also more interconnection points in the virtualized environment [23]. The majority of the new security protection programs that are being introduced to the market by different vendors are basically concentrated on hypervisor. Hypervisor is basically a software program and therefore similar to any software, it has all the traditional software bugs and the security weaknesses that are possible for any

software have these weaknesses. These are some reason that security in the virtual environment in more complex than a non-virtualization environment.

There are different types of virtualization technology such as full virtualization, para virtualization, application virtualization, hardware support virtualization, resource virtualization and container virtualization that are explained below [24]:

2.2.1 Full virtualization

In this method, multiple logical instances of completely independent virtual machines are simulated by hypervisor, with its own virtual resources. Input-output ports and DMA channels are the virtual resources that are used in this approach. Consequently, every operating system that is supported by underlying hardware can be run by each virtual machine. It's difficult to manage the full virtualization due to the overhead of handling these operations CPU. Nevertheless, the VM environment is able to provide "full virtualization" if it can offer an adequate illustration of the underlying hardware for the guest operating systems to run without modification. Full virtualization is shown in Figure 2.3:

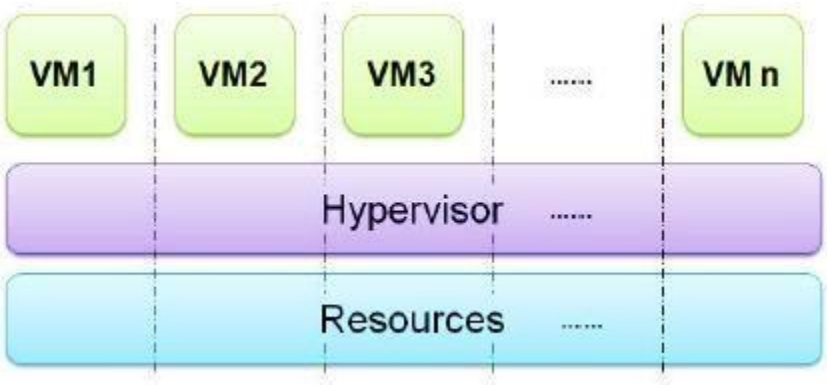


Figure 2.3: Full virtualization

2.2.2 Para virtualization

This approach is one of the first approved types of virtualization which is still utilized in a wide range of application. Para Virtual Machine (PVM) relies on special kernels and drivers and therefore does not need any particular hardware for realizing the virtualization. By using these special kernels and drivers choosing the operating systems are more limited. Particularly, an OS that are adjustable to work with hypervisor must be used by PVM. As shown in Figure 2.4, in

PVM one virtual machine which is the main VM has access to the resources and can allocate resources to other VMs.

Since a main VM is controller in PVM and this main VM is in the same layer with other VMs, so access to the main VM (Dom 0) is much easier; however, if we have Full virtualization, access to hypervisor and getting system's control is more difficult because VMs and controller are not in same layer in this type of virtualization. Hence Full virtualization is more secured than Para virtualization.

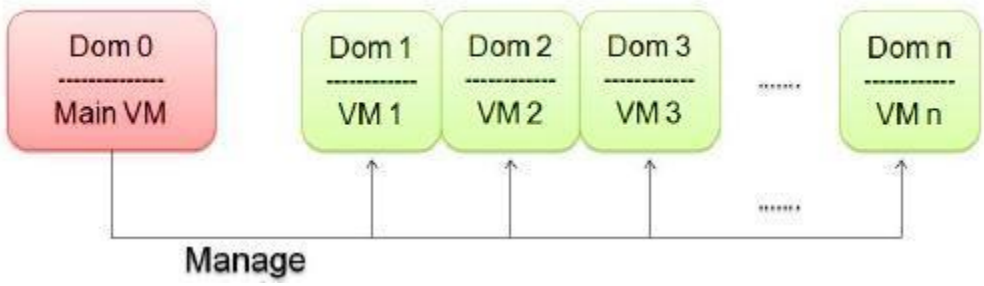


Figure 2.4 : Para virtualization(Xen)

2.2.3 Application virtualization

In this approach, a server application can be run locally using the local resources without any need to install the complete application on user's computer. These virtualized applications can be run in a small virtual environment having only the necessary resources for execution of the application [25].

2.2.4 Hardware support virtualization

Hardware Virtual Machine (HVM) considered as the lowest level of virtualization. In HVM, in order to trap privileged calls from guest domains, a particular hardware capability is required. In this method, it is likely to have a fully virtualized machine without having any particular operating systems or drivers on the guest system. The hypervisor locates in the root mode and is in charge of resource allocation and I/O device interaction. However all the guest operating systems locate in the non-root mode and therefore they call out for the hypervisor to process their requests for resources. "Hyper calls" is a special virtualization instruction which is used to process the request of the guest operation systems [26]. The majority of recent CPUs are built with HVM capabilities, normally known as virtualization extensions [27].

2.2.5 Resource virtualization

Resource virtualization which is also called Storage Virtualization is the process of virtualizing system specific resources such as storage[25]. A variety of methods are available for resource virtualization process. The typical approaches are as follows [24, 28]:

- Combining several separate components into a greater resource pool.
- Combining multiple discrete computers in grid computing or computer clusters in order to have a large supercomputer with massive resources.
- Creating a number of small and easily accessible resources by partitioning a single resource of the same type such as disk space.

2.2.6 Container Virtualization

Container virtualization also known as OS-level virtualization creates multiple secure containers on a single machine. In this approach, a user can share a single kernel among several containers so that they can utilize computer resources securely with a minimal interference from other containers. According to Padala et al.[29], this method has the lowest overhead among all the existing virtualization techniques.

2.3 Cloud standards

In this section we want to speak about some definition in cloud computing environment:

Vswitch: Vswitch in virtualization network is like a normal switch in traditional network. All traffic in virtualization network should come to Vswitch, and Vswitch determine where is the destination of packets. Place of Vswitch is in hypervisor[30].

Virtualization layer manager (hypervisor manager): it is a manager in lower level. As we see in section (2.2) virtualization is an important concept in cloud computing. Hypervisor management can manage all activities related to hypervisor. For better understanding I explain more with an example; Xen is a kind of hypervisor. Xen has a non-graphical interface, so this is difficult to use and manage hypervisor directly and also directly use of Xen command for managing is not good for security, so we have a software that called Xen manager. Xen manager is a graphical interface that can manage all activities related to Xen. Xen manager can create

VMs, check traffic and can check everything about VMs, create account and everything for a hypervisor in a data center. Xen manager is a hypervisor manager [31].

Cloud manager: there is a cloud manager for a cloud provider. Cloud manager is a higher level manager than hypervisor manager. A cloud manager can manage all hypervisor managers in a cloud provider company. Openstack [32] is a cloud manager.

2.4 Cloud security

The security of the cloud or the virtualization environment security is more difficult compared to the old networks due the availability of more systems to protect, more possible points of entry, more holes to patch and also more interconnection points in the virtualized environment [1] .

A Cloud system at least must contain 5 important features to achieve sufficient security [33, 34]: availability, confidentiality, data integrity, control and audit. Though, it is difficult to achieve all the five features together.

2.4.1 Availability

Availability means, costumers have access to the cloud resources anywhere and anytime[33]. The availability in the cloud technologies can be increased through widespread internet-enabled access, although the user will remain dependent on the robust and timely provision of resources. Availability can be supported by capacity building and good architecture by the provider, while defined contracts and terms of agreement are considered as important supporters as well. Availability is a big concern in cloud computing because in cloud providers side, provide more availability means more reliability and more credit and in customers side, get services from a company with more availability means have access in their data from anywhere and in any time that they (customers) want.

Availability can be improved by using some techniques like Migration (specially “live migration”) and Load balancing in the cloud environment.

Migration is a technique which is used in cloud computing environment when a server has a lot of tasks to do or when the requests in this server are very high and pass a threshold. Using this technique, the server can migrate the tasks to other server in the cloud which is connected to the previous one and can do these tasks. For keeping a good availability, it is desirable to allow the

application of a set of resources to be transferred to other resources without the resources being stopped; as the user should not know the problem is happened. This kind of transfer called Live Migration. Live migration is a big concept for research. The strategy of live migration of multiple virtual machines is made with different methods of resource reservation[35]. There are examples of Live Migration algorithm in [36, 37].

Load balancing is another factor that has a significant impact on the performance of cloud computing environment [38]. Good load balancing results in a more efficient cloud computing and consequently the user satisfaction is improved. A variety of load balancing algorithms are available such as an algorithm for load distribution of workloads among nodes of a cloud by the use of Ant Colony Optimization (ACO) that is provided by K. Nishant et al.[39] and a model of load balancing for the public cloud based on the concept of partitioning clouds with a switching mechanism to choose different strategies for different situations is provided by Gaochao Xu et al. [38].

2.4.2 Confidentiality

Confidentiality is defined as the ability to keep the users' data safe and secret in the Cloud systems[34]. For many users, the confidentiality of the cloud systems is a main barrier to join in cloud and many users believe that their business sensitive data will never be in the cloud "in the article entitled" Above the cloud [40].

Accordingly, in order to attract even more users to the Cloud, the confidentiality is a fundamental requirement.

Storage encryption is an option to improve confidentiality [33]. For instance by encrypting the data before put it in a Cloud; you make them even more secure than unencrypted data in a local data center. However, it can cause some problems itself. If we want to storage encrypt data in the cloud, the encryption needs to be done in our system. We also should download our data from cloud storage and decrypt them in our own system; therefore, we require a computer or a system with a processor and hardware and it is contrary to the cloud benefits. Cloud environment wants to go in a way that the users can have access to data and their computing work with a simple device that only have a browser; something like a very simple mobile phone.

2.4.3 Data Integrity

In the cloud system, data integrity is defined as protecting information integrity (e.g., to protect them from lost or modification by unauthorized users). [41]

As data is the base for both cloud computing providers and users it is very crucial to protect it. Consequently keeping data integrity is a fundamental mission which is expected to be attained by techniques such as RAID-liked strategies[42], digital signature[43].

2.4.4 Control

In the Cloud system Control is defined as the regulation of the usage of the system, including the applications, their infrastructure and the data. In the Cloud Computing system, distributed computation is always involved on multiple large-scale data sets across a large number of computer nodes. Furthermore, all internet users can contribute their individual data to the Cloud Computer systems located on the other side of the Internet, and start using them. Storing all these personal data in the environment of cloud computing may result in the system users of cloud computing being faced to many threats to the personal data[41]. Users are concerning about privacy of their data because user don't know where is their information and they should put their information in a place that is not in their control.

2.4.5 Audit

Audit means watching what occurred in the Cloud system. To provide facilities with the ability to watch what happened in the system, we can add auditability as an additional layer above the virtualized operation system (or virtualized application environment) hosted on the virtual machine.To do so, three major attributes should be audited:

- **Events**: All factors that affected the system availability including the state changes.
- **Logs**: Comprehensive information about users' application and its runtime environment.
- **Monitoring**: It has to be limited to the reasonable requirement of the Cloud provider for running their facility.

As a result of adding this new feature (i.e., adding the auditability as an extra layer in the virtual operation systems) the Cloud Computing developers have to focus on providing virtualized capabilities instead of specific hardware[41]. Actually audit is one of the cloud manager task that can provide more security in cloud computing.

2.5 Security challenges

Architecture's problems and attacks in cloud environment will result in losing one or some of the above concepts in cloud computing or virtualization environment. Cloud computing introduces new concepts such as multi-tenancy that creates new challenges to the security community [44, 45] and introduces additional security implications [46, 47].

Some of the Security weaknesses in virtualization environment and cloud computing are listed below:

2.5.1 Rootkit Attacks

A rootkit is a type of malicious stealthy software that provides privileged access to the system and gives the attacker the ability to hide the existence of several malicious processes or programs in order to make them undetectable[48, 49]. As malicious processes can make a lot of problem for cloud so Rootkit can attack to all availability, confidentiality, data integrity, control and audit [50].

2.5.2 VM Escape

While virtual machines are permitted to share the resources of the host machine, they still are able to provide isolation between VMs and between the VMs and the host. Specifically, because of the way the virtual machines are designed, a program running in one virtual machine cannot monitor, or communicate with programs running in either other VMs or the host. But in real cases isolation is compromised by the organizations. The flexible isolation is configured to meet the organization requirements which exploit the security of the systems [49]. A VM that has access to the host can do whatever it wants, so this is risky for all security objectives especially for availability, confidentiality and data integrity [26, 46, 47].

2.5.3 VM monitoring from another VM

Isolation plays an important role in virtualization. However, it can be considered as a threat if one VM without any difficulties is allowed to monitor resources of another VM [51]. It is dangerous for confidentiality goals [52].

2.5.4 Guest-to-Guest attack

Isolation can become a threat to the environment if it is not carefully deployed. It is very crucial that isolation is wisely configured and maintained in a virtual environment in order to guarantee that none of the applications running in one VM has access to the applications running in another VM. On the other word, isolation should be strongly maintained to ensure that break-in into one virtual machine can never provide any access to virtual machines in the same environment or to the underlying host machine. It is known as guest-to-guest attack [23, 51, 53].

Confidentiality and data integrity even availability may take effect in these attacks.

2.5.5 Denial of service attack

Denial of service (Dos) is one of the most common intrusion approaches and often causes countless economic losses and impact. A non-legitimate user tries Dos attack to degrade or deny resources to legitimate users. DoS attack has many results for instance: the system may work very slowly or the user may be not able to connect the network server. Denial of service is attacked to the availability of the server.

The Distributed DoS named as DDoS is the most popular method of DoS attack as it is able to cause more serious effects easily and rapidly.

Figure2.5 shows DOS and DDOS attack in old system and DDOS attack in cloud computing systems. It is shown that the DDOS attack is more complex than DOS one because DDoS attack consists of three basic elements: 1) The chief or the attacker, 2) Host machine (zombie) and 3) victim machine. At the first step of attack, main attacker (chief) finds many host machines (zombies) and implements some software on them to have a communication with them for future (attack time). The main step of attack happens by the host machine with an order from the chief. An attack which is a combination of many different origins is far more difficult than the attack which contains only one source.

In cloud computing system, the situation is even more complicated because in cloud computing there is a distributed server (figure 2.5 (below)) and it causes a bigger problem. Virtualization, multi tenancy and the use of share resources are some of the reasons that make DDOS attack much more serious and more difficult in cloud computing system compared to the old network (non-virtualization environment).

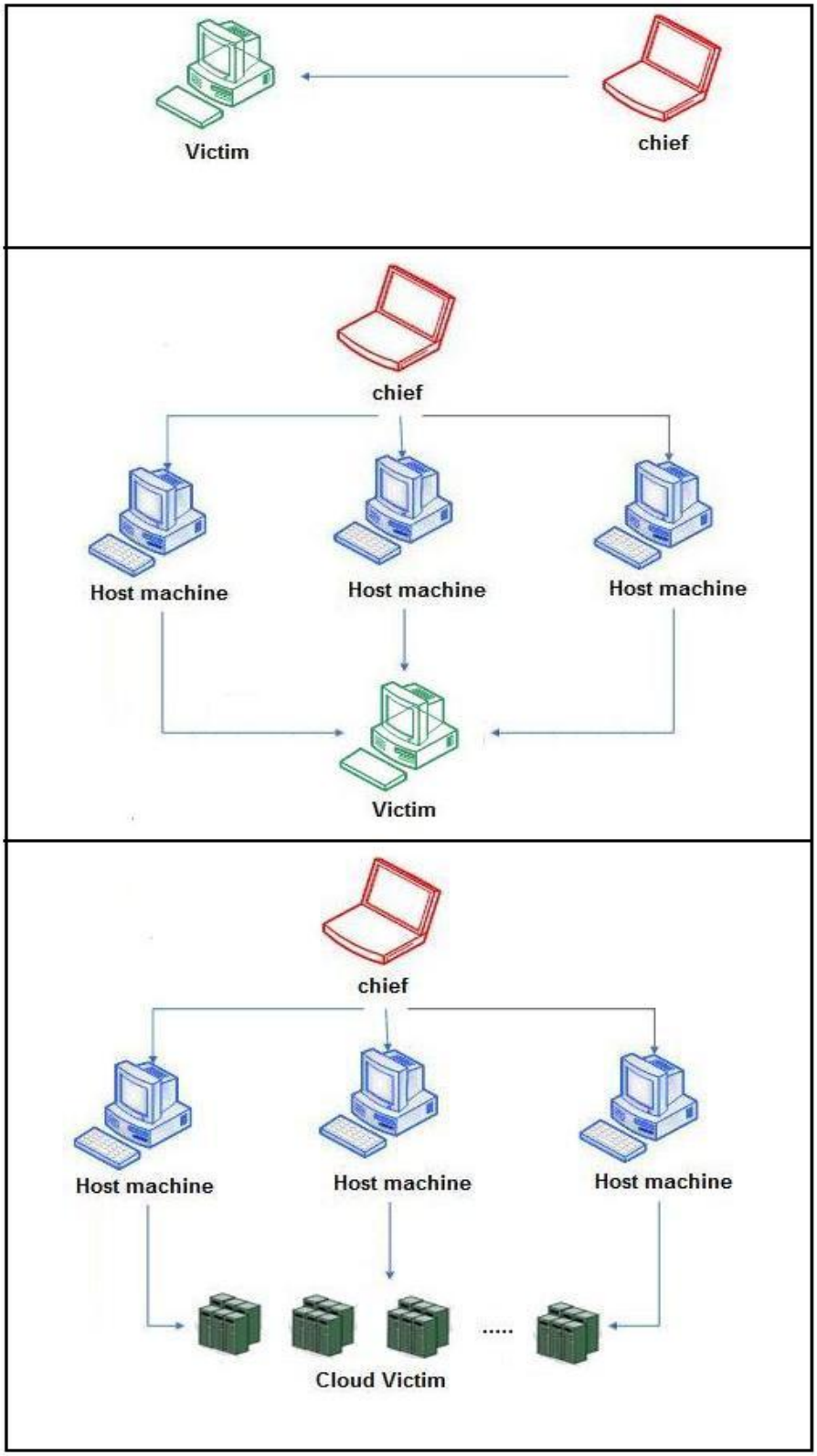


Figure 2.5:DoS attack (above) and DDoS attack (middle) in old system and DDoS attack in cloud computing system (below)

Also virtualization technique itself is causing overload on the server [54] so even a small attack in cloud computing environment can disable the server.

Here, a quick review of some type of DOS attacks is provided:

2.5.5.1 Application Layer DDOS

Application layer DDos attacks is a type of DDOs that happend on the application layer. Attacker uses vulnerabilities of protocols that used by applications in this type of attack[55]. The most important victim for this type of attacks are web services[56]. Application layer distributed denial of service (app DDOS) attacks usually use less bandwidth and are stealthier in nature when compared to volumetric attacks. However, they can have a similar impact to service as they target specific characteristics of well-known applications such as HTTP, DNS, VoIP or Simple Mail Transfer Protocol (SMTP) [57].

X-DoS and H-DoS are two type of web server attacks and most attackers are more inclined to use of these web based attacks because their implementation are simple than others [58, 59].

Extensible Markup Language (XML) DOS (X-DOS) occurs when an XML message is sent to a Web Server or Web Service with malicious content to use up all their resources[60].

Hyper Text Transfer Protocol (HTTP) based Denial of Service (H-DOS) attacks can send randomized HTTP requests to the victim web server to exhaust its communication channels [61].

2.5.5.2 UDP Flood

The User Datagram Protocol (UDP) flooding attacks exploit UDP's connectionless mechanism and its limitation in responding to the request. Ports on a remote host will be attacked by flood of many UDP packets, it force the host to constantly check for the application listening at that port; since there is no application, so a response will be sent that there is destination Inaccessible packet[62].

2.5.5.3 TCP SYN Flood

Transmission Control Protocol (TCP) SYN Flood is one of the simplest and most common attacks seen on the Internet [54]. Several requests by attackers for huge instances of making connection, lead a denial of service attack. This attack benefits from the number of allocated

resources by a server in order to perform a 3-way handshake[54]. Generally, when a request is sent to the server for getting a connection, this request is estimated and authenticated, and then the user can pay to get the service. The flooding attack is occurring in the authenticity step. So there is no need to pay for that. An attacker overloads a victim with a lot of connection requests so that it cannot respond to legitimate requests, since the IP address of attacker is false. For this attack, many TCP SYN packets are sent to the victim. For each new TCP connection the victim allocates buffers and consequently a SYN-ACK is transmitted in response to the connection request [54].

To have a better understanding of these kinds of Dos attacks, I give an example in the real world. Imagine that there is a store with one seller and a lot of clients. Assume that each client comes in the store and asks about the prices of the merchandises. They just ask without buying anything. When the seller wants to answer costumer's question, he/she (costumer) escape. If for example 1000 clients come to the store, seller can't answer all their questions at the same time and the seller's mind is crashed.

If we bring this example to the network systems we can put hackers instead of clients, packets instead of merchandises and server instead of seller.

Base on [54], the steps that are involved in the TCP SYN Flooding attack are summarized here:

- An attacker sends a large number of service requests with false address.
- The server sends a response message back to the sender.
- The server waits for the response information from the user.
- Since the addresses have been forgotten, the server can't get any information and must wait for a long time and the connection will be cut off with overtime.
- The resource allocated for this request cannot be released.
- If the request number is very large, the server resources will be used up finally. So the new user can't get the service and the attack is completed successfully.

2.5.6 Economic Denial of Sustainability (EDoS) (Special DDOS attack for cloud computing)

Economic denial of sustainability is a scenario in which the customer pays extra to the cloud provider because of the attack. According to the general definition of the National Institute of Standards and Technology (NIST) one of the most important benefits of using cloud computing is to have On-demand services[3]. It means that cloud customers don't need to buy whole resources and infrastructure for the first time. The method of payment in cloud environment is pay per use[4]. Based on this benefit of cloud computing, there is a new kind of DDOS attack which is called Economic Denial of Sustainability (EDoS)[5]. Since cloud environment is elastic and works based on the pay as you use model, additional resources are easily available; but the customer has to pay extra for them. Since EDoS attacks are considered for economic issues that thes type of attacks bring to cloud, so different types of DDoS attacks that we mentioned above can count as an EDoS attack. Also, because in this situation DDoS attack is transferred to the hypervisor, it can be dangerous for the Hypervisor and finally for the entire cloud. One way to prevent this problem is to limit the resource allocation[9]; however, by doing so we actually limit some of the most important advantages of the cloud computing .

2.6 Related work in Defence and Detection algorithm against DDOS and DOS

Both DoS and DDoS attacks are serious threats to the Internet. Consequently, it is necessary to have a more intricate mechanism to determine the malicious traffic from the legal ones. Different detection and defence algorithms have been introduced in the literature [58, 63-74] and etc, for cloud computing and in traditional system.

2.6.1 Defense and Detection Solutions for Cloud Environments

Lanjuan Yang et al. [63] adopted an approach based on a traceback Service Oriented Architecture (SOA), to identify the source of a DDoS attack on the cloud. They proposed a detection approach called SBTA (SOA-Based Traceback Approach) to traceback and identify the source of DDoS attacks. They also offered a filter called Cloud-filter. In the proposed architecture, firstly all

service requests go to SBTA for marking, then if the packet is identified as normal, the services will be provided to the corresponding customer.

Some of the current attacks that attackers may initiate as HTTP and XML have been reproduced in [64]. They also proposed a solution for trackbacking based on their Cloud TraceBack (CTB) approach to find the source of these attacks. Furthermore, they introduced a back-propagation neutral network, called Cloud Protector, which was trained to detect and filter such attack traffic. The authors of [64] used their previous work on the service-oriented traceback architecture (now called Cloud TraceBack) to defend against X- DoS [58, 66] in the field of cloud computing.

Jin Wang et al.[65] focus on the application layer DDoS (app-DDoS) attacks, and a new relative entropy based app-DDoS detection method was proposed. They conducted a 3-stage strategy: (1) the click ratio of the web object was defined and the cluster method was used to extract the click ratio features; (2) with the click ratio features of the learning stage, the relative entropy was used to detect the dubious sessions; (3) experiments based on the real traffic were conducted to validate their detection method.

Ang Li et al [67] presented a new algorithm for detecting anomalies in a large data center base on analyzing IP (source and destination) structure and design a lightweight anomaly detection algorithm that can be used for very large data centers. The authors used the analyzing real-world traffic traces in order to show several characteristics of the IP address distribution observed in large data centers for cloud computing services. In this approach, they revealed a stable concentration on selected bytes of both source and destination IP addresses. Eventually they could derive centroid based statistics to characterize data center traffic. Basically, the concentrated distribution of octets in the IP addresses is a common characteristic of the traffic flows in data centers and this is the feature that has been used in this article as a main parameter in their algorithm. The presented algorithm in [67] consistently diagnoses the abnormal traffic from normal ones, and does so in a short time (but not always) with a low false alarm rate. This algorithm is very sensitive and has a low run-time overhead, so it is good to use it in large-scale data centers. In this algorithm an adaptive threshold has been used instead of a fixed one. Because, it is not possible to precisely distinguish a normal situation from abnormal by using a fixed threshold. But this detection algorithm works based on IP (source and destination) address and in the case of DDOS Flooding attack we don’t have a true source IP address; therefore this

algorithm is not able to detect this attack. On the other hand, based on our knowledge in some DDoS attacks, attackers use a special script in data part of IP address that changes the header uninterrupted. The approach presented in this paper is not useful in detecting these kinds of DDoS attacks because this approach is based on IP address and IP isn't a good parameter for detecting this kind of attacks.

In [68] authors provided an approach to monitor the resource usage by using VM for Kernel-based Virtual Machine (KVM) system. Generally, it is very important to know how much resources are used by each VM to recognize the VM and subsequently the user who uses high percentage of the resources, and to limit providing the service to this user if needed. In this paper, the authors presented a performance monitoring tool for KVM based virtual machines, which measures the CPU overhead incurred by the hypervisor on behalf of the virtual machine along-with the CPU usage of virtual machine itself.

They assumed that the overhead is negligible as it is less than 1 percent on the host system. This approach can be used for resource provisioning to support performance associated QoS requirements, identification of bottlenecks during VM placements, and resource profiling of applications in cloud environments. Also presented approach in [68] gives separate hypervisor usage per VM.

This approach is good enough for a normal situation in virtualization; however it is not able to distinguish between a legitimate user and an illegitimate one. The legitimate user may have a high percentage of CPU, but we cannot be certain that this user is an attacker.

Ryan Shea et al. [54] tried to find a solution for Dos attack by using a threshold and modifying it in the interrupt request. However, using a threshold may not be a good solution because the small threshold improves the sensitivity, but increases the possibility of introducing false positives. On the other hand, a larger threshold is likely to reduce the sensitivity of detection. Basically, in normal situations, we might have a large load in the network without any attack; however, using the threshold this situation is considered as an attack.

In addition to the contributions presented above, good detection algorithms have been proposed for non-virtualized environment; some of them are summarized hereafter.

2.6.2 Defense and Detection Solutions for non-Cloud Environments

In [69] a network monitoring tool called AGURI is presented. It can detect DDOS attacks using collected traffic patterns in long-term monitoring network traffic. For detecting the flooding attacks in this algorithm, they needed to know information about normal situation in network (when attack does not appear). The algorithm requires a large amount of traffic data. Simple Network Management Protocol (SNMP) is one of simple mechanisms for collecting the data from the routers and switches. So they use of SNMP for this goal.

In [70], batch sequential and adaptive sequential methods for rapid detection of DDOS attacks are introduced. In this approach, with statistical analysis of data on different layers of the network, they try to find a sudden change in the traffic. The idea is that DDOS attacks make a sudden change in the traffic model in comparison with a normal traffic. Authors provide a detection method in this paper that employs statistical analysis of data from multiple layers of the network protocol for detecting very subtle traffic changes, which are usually for DOS attacks. Threshold of test statistics is used by both the batch and adaptive sequential methods to achieve a fixed rate of false alarms. In addition, these methods can detect combinations of different types of DOS attacks.

In [71], MULTOPS (MUlti- Level Tree for Online Packet Statistics) data structure is presented for detecting DDOS attacks. By using this data structure, the input stream into the network will be compared to the output data stream. If any asymmetry is found between input and output, it will be considered as an attack. Routers or network monitors are enabled to detect ongoing bandwidth attacks by MULTOPS and with using a simple heuristic based on disproportionate difference between the packet rate going to and coming from a host or subnet. This is based on the assumption that, the packet rate of traffic going in one direction is proportional to the packet rate of traffic going in the opposite direction in normal traffic situation.

In [72, 73] network traffic that is expressed in the rate of TCP signs and rate of different contracts has been analyzed and it has been shown that when the attack occurred, the rates are changed. In addition to detecting DDOS attacks, a set of state actions generated by machine learning algorithms has been evaluated in a simulated environment. A network traffic analysis mechanism is provided in [72] which computes the ratio of the number of TCP flags to the total number of TCP packets. Authors compile a pair of the TCP flag rates and appear or not appear of the DDoS

attack into state-action rules with using machine learning algorithms base on the calculation of TCP flag rates.

In [74], a method for the detection of DDOS attacks is discussed which is fundamentally based on the information theory especially Kolmogorov complexity. This approach assumes that the attackers in DDOS attack are trying to flood the target machine by producing similar packet from a different source. It is alleged that by using a method based on the "Kolmogorov complexity" the packets with similar patterns can be detected. Authors believe that one of the important points in this algorithm is that it does not require special filtering rules and hence it can detect any type of DDoS attack. They provide an algorithm that is shown to perform better than simple packet-counting or load-measuring approaches.

All of these previous contributions tried to detect DDOS attacks in non-virtualization environment but as we discussed the problem in the virtualization environment is much more complex and bigger than non-virtualization environment. Nevertheless we think that with use of some technique and a good architecture design, we can leverage a combination of detection DDOS attack algorithms from non-virtualization to cloud. Although, the main concept and definition of DDOS attack is same in cloud and non-virtualization system with small substitutions (for example there is resources sharing and on demand resources in cloud computing). The differentiation point is in efficiency of this attack that is much more in cloud computing as discussed before and also another point is cloud architecture is different from non-virtualization systems.

2.7 Related work in detection and defense against EDoS attacks in cloud computing

Some approaches have been introduced in the literature on how to detect, defense and mitigate EDoS attacks in cloud computing environment.

Sqalli et al. [75] proposed an approach called EDoS-Shield to mitigate EDoS attack in Cloud Computing. They divided the received requests from the user side into two categories: the legitimate request and the generated by bots request. They used a verifier and sent the first request from a new IP to this verifier. The verifier does a verification process and decides to put this address either in the white or the black list. These lists are related to legitimate requests and

bots generated requests, respectively. The subsequent requests that are from an IP address in the black list will be blocked by a virtual firewall and the subsequent requests from an IP address in the white list can connect to the cloud services to get services that they want.

The proposed work by Naresh Kumar et al. [76] provided an architecture to mitigate the web service EDoS attacks. In this work a crypto puzzle (Client puzzle) has been generated to identify a legitimate user. The customers must solve this puzzle to have an authorization for using the cloud services. Their work is based on the service provider assessment about the state of the system, which can be normal or suspicious. Normal and suspicious states are depending on the Server load and the Bandwidth load. Using this information, the architecture decides about the severity of the puzzle.

Enhanced DDoS- Mitigation System (Enhanced DDoS-MS) is a framework presented by Alosaimi et al. [77] to mitigate EDoS attacks in cloud computing. This framework identifies legitimate users from malicious ones by testing the first packet received from the user by using a Graphical Turing Test (GTT). Also they utilized an Intrusion Prevention System (IPS) to detect malware in the contents of packets. Crypto puzzle and white/black list approaches were used to mitigate EDoS attacks in this work similar to the previous articles that we discussed [75, 76].

A cost effective EDoS attack mitigation framework for E-Commerce applications in cloud environments has been provided by Masood et al. [78] which is called EDoS Armor. This paper provided a multi-layered defense system; at the first step, only a limited number of valid users are allowed to connect to the application to be less bothering. Next, a browsing behavior-based learning mechanism is used and the priority to the users is allocated on this basis. This priority value determines how the resources are shared among the users; the higher the priority the more resources are allocated and the lower the priority the fewer resources are assigned.

An approach to control virtual resource access to mitigate EDoS attacks against cloud infrastructures is provided by Baig et al. [79]. In this paper an approach has been provided to selectively control the user requests for a service within the service provider. In this approach incoming requests from the user are classified into normal and suspicious requests. Afterward, further analysis is implemented to guarantee that those normal and legitimate users have the priority of cloud service access; however, users which are in the suspicious class have a lesser priority to the service access, until they are removed from the suspect list.

Al-Haidari et al. [80] provided an algorithm that uses Time To Live (TTL) field of the IP header to detect and mitigate the spoofed IP EDoS attacks. The authors made a use of the white/black list approach in their work. They used a verifier and a threshold system to classify the entering packets to the normal and the suspicious ones and to collect source IP addresses along with the amounts of TTL in the whitelist and the blacklist for the normal and the suspicious packets, respectively.

Koduru et al. [81] proposed a detection approach for EDoS. They used the Time Spent on a Web Page (TSP) to detect HTTP EDoS attacks. The main idea of their work came from the fact that the TSP in the attack situation is different from the mean TSP of a web page in the normal situation. They calculated TSP and the Mean Absolute Deviation (MAD) of TSP and showed the results in a graph. But the provided approach in this paper doesn't work automatically and a cloud administrator needs to always monitor the graphs. Also this paper provided a solution for detecting only HTTP EDoS attacks and cannot detect other kinds of EDoS attacks.

As shown in this review, the majority of the papers in the field of EDoS attacks have focused on finding a solution for the defense and mitigation of only one type of EDoS attacks in cloud computing. Although, in general, there are a few works on detecting this kind of attack as well. In the next chapter we will add new metrics to these structures and will propose a new model for detecting different types of EDoS attacks in cloud computing and we want to find out if using one inclusive framework for simultaneously detecting different types of EDOS attack is more accurate than using a separate framework for each attack.

2.8 Open problem

DDoS attacks and a special type of these attacks that called TCP SYN flooding attacks are common and popular attacks in cloud computing. EDoS which is a new and special type of DDoS attack can also influence Cloud through a DDoS and make economic problem for cloud costumer and provider. These attacks, specially when combined, cause a big problem against availability.

In order to have a secure system against these attacks, three algorithms are required for the following purposes:

1- Detecting the attack

2- Defense (based on a good detection) against the attack that took place in the system

3- Preventing the future problems.

The first important step in confronting the attack is to detect it. To obtain a good result in both defense and prevention against this attack primarily we need to have a reliable system for detecting the attack with an acceptable and high percentage of reliability.

There are a lot of algorithms in traditional (non-virtualization) system for detecting DDoS attacks such as presented algorithm in [69-73]; However, virtualization technique itself is causing overload on the server [54] and accordingly even a small attack in cloud computing environment can disable the server. Therefore, the traditional algorithms which work well in non-virtualization system are useless in cloud environment if we don't apply the required modification.

The majority of existing algorithms for detection DDoS attacks in cloud computing environment work based on packet information (for example IP address); However, a big challenge in this case is that the IP address in DDOS attacks can be false. Furthermore, the packet in DDoS attack is like packet in normal situation and the only difference is that there are too many packets in DDoS attack; thus packet cannot be a good area to work on for detecting DDoS [69, 72]. Consequently the present packet based approaches are not reliable and powerful enough for DDoS detection.

On the other hand,the current algorithms for EDoS attacks are useful only for one attack and to our best knowledge there is not any global model to detect all types of EDOS .Moreover, they have mainly focused only on finding a solution for the defense and mitigation of EDoS attacks in cloud computing.

Accordingly it is necessary to design a novel, powerful and inclusive algorithm which works for all types of DDoS and EDoS attacks in cloud computing. Working on "traffic and resource usage anomaly detection" seems to be a promising approach for this purpose. It's mainly because of the fact that even if attackers in DDoS attack can make packets like normal packets but the traffic generated in DDoS attack isn't identical to traffic in normal situation. Additionally, in the time of attack proportion of resource usage will be different from this proportion in the normal situation. Therefore, investigating the features of resource usage in DDoS attacks can result in a better detection.

These features vary depends on the type of attack and accordingly in order to design an inclusive and general algorithm we need to add all features of all attacks to our model. Consequently, using this work, cloud providers can understand which type of attacks happened in their system and can implement a true strategy to defense against that attack.

Features of one kind of attack play an important role to detect another type; consequently we can expect much more reliable results using a global and all inclusive algorithm compared to the algorithms which consider only the features of one attack. Accordingly, it is obvious that we don’t compare the attacks themselves with each other. Instead, the virtual machines which are allocated to the victim server will be compared.

CHAPTER 3 METHODOLOGY

As discused in chapter 1, our aim is to detect DDoS and EDoS attacks in cloud computing by working on traffic and resource usage anomaly detection.

Our main idea for detecting these attacks is that, even if attackers in DDoS and EDoS attacks can make packets like normal packets but the traffic generated in DDOS attack isn't like traffic in normal situation. Also the proportion of resource usage is completly different in case of attack compareed to the normal situation. Therefore we investigate the samples of traffic and resource usage in cloud computing servers in order to find a proper approach to detect these 3 types of attacks (HTTP attack, Database attack and TCP SYN Flood attack).

Our ultimate goal is to introduce a global and inclusive algorithm which works perfectly for HTTP attack, Database attack and TCP SYN Flood attack no matter what time each attack happens. Consequently, instead of using three different algorithms for detecting these attacks we will be able to use a unique and generalized algorithm. This way, we get much more reliable results compared to the algorithms which consider only the metrics of one attack because specific metrics of one kind of attack can play an important role to detect another type.

This section describes our methodology to address the research problems and achieve the mentioned objective. In order to detect EDoS and DDoS attacks in cloud computing, we will discuss about the information of normal traffic and resources in section (3.1). Section (3.2) explains our monitoring and sampling module. Section (3.3) describes the attack identification; section (3.4) explains the metrics which will be used in detection procedure; section (3.5) describes the change point detection algorithm; section (3.6) provides our algorithm to detect different types of DDoS and EDoS attacks and finally section (3.7) is a conclusion for this chapter.

3.1 Extracting the information for the normal traffic and resources

At the very first step, we will need to have a normal traffic and resources from the hypervisor. We must first generate this traffic intentionally and collect the samples. The amount of the samples must be enough to cover the various traffic situations at different hours of the day. These traffic samples must cover heavy and light traffic times as various services may be provided to customer by the cloud provider depending on the time. We want to analyze this traffic and the

resource usage in normal situation to extract some thresholds and investigate the behavior and proportion of resource usage in normal situation and extract some information. With this information, we can verify if there is a sudden change in the traffic and resources (as it will be explained in details in section (3.3)).

3.2 Monitoring and Sampling Module

We will need to generate HTTP attack, Database attack and TCP SYN Flood attack traffics in order to collect the corresponding samples and investigate the behavior of resource usage.

As we discussed in chapter (2), the virtualization is the main concept of cloud computing and every service in cloud computing is located in a virtual machine. For example, the database is placed in one virtual machine and the web server is located in a separate virtual machine. Therefore when customers want to put their services in the cloud, they must order a set of VMs (we consider the infrastructure as a service). However, as discussed earlier, cloud computing system is a pay-as-you-go service, so customers have to pay for the amount of VM that they used. They can apply for more resources whenever they want.

On the hypervisor side, we monitor all the events and the requests from each VM to extract some metrics.

In our work, every time a virtual machine applies for more resources and sends its requests to the hypervisor we can have a sampling and monitoring system. That way, we can trace the kernel system of VM which sent a request and VMs which communicate with that VM. We will then be able to get the samples from the traffics of VMs for our analysis.

To achieve this goal it is required to design a “Monitoring and Sampling” module as a part of the targeted DDoS detection architecture. Hypervisor is the best place for positioning this part because all the traffics for all virtual machines come to the hypervisor. Therefore the traffic in the hypervisor consists of a combination of traffics in all VMs and consequently hypervisor can have access to each VMs resource to extract metrics. Figure 3.1 shows Detecting and monitoring system.

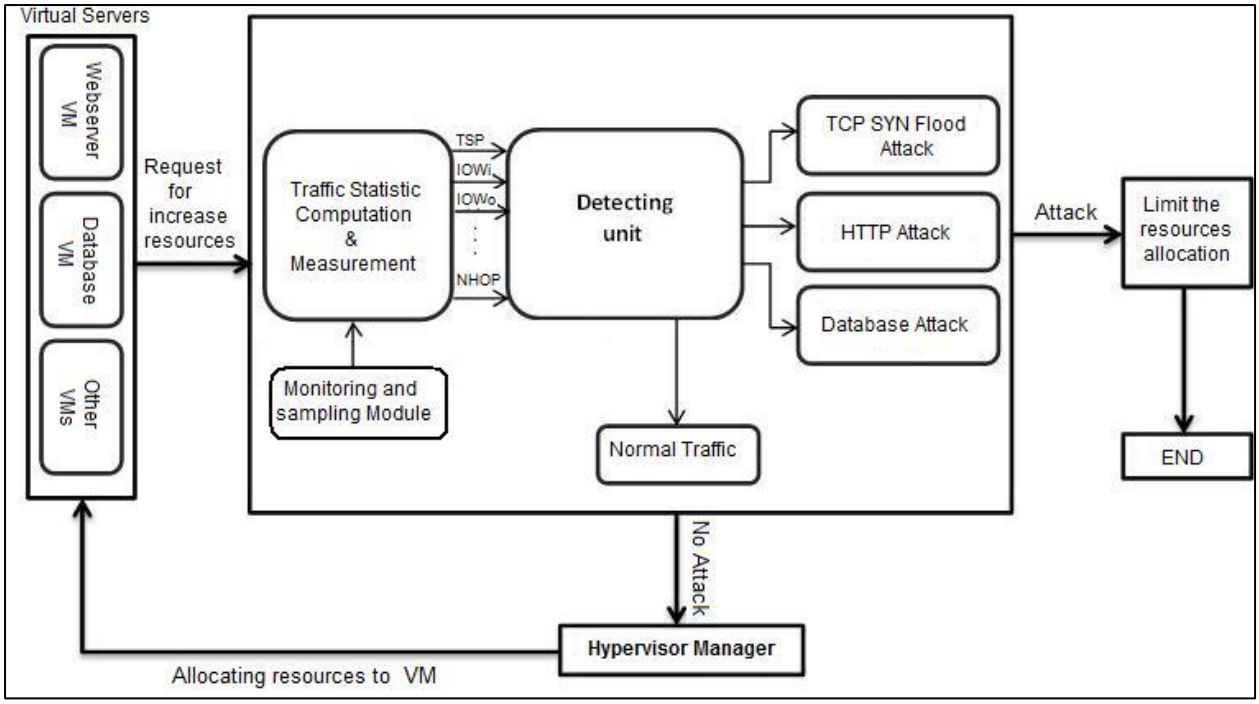


Figure 3.1: Detecting and monitoring system

3.3 Attack identification

Our main approach to detect HTTP attack, Database attack and TCP SYN Flood attack is designed based on the fact that even if attackers in DDoS and EDoS attacks can make packets like normal packets but the traffic generated in attack situation isn't identical to traffic in a normal situation. Also the proportion of resource usage is completely different in case of attack compared to normal situation. Accordingly each attack can be identified by specific features which are related to the traffic and resource usage in in that attack.

Furthermore, as discussed in chapter 2, every service in cloud computing is located in a virtual machine. For example, the database is placed in one virtual machine and the web server is located in a separate virtual machine. Different types of attack target different virtual machines as follows:

- **EDoS attacks including HTTP attack and Database attack:** In HTTP attack Webserver virtual machine will be targeted by attackers. In case of Database attacks the virtual machine allocated to Database receives many database queries for each HTTP request [7, 8].
- **DDoS attacks including TCP SYN Flood attack:** TCP SYN Flood attack occurs in webserver virtual machine.

To design our model, we need to investigate the behaviour of resource usage in virtual machines which are in relationship with the victim virtual machine. For this purpose we collect features which are related to traffic and resource usage in each attack. These features are the metrics of our profile and will be explained in details in the following section.

It is important to notice that we don't compare the attacks themselves with each other. However, instead of using a separate algorithm for detecting each attack we will introduce an inclusive and generalized algorithm which contains the metrics of all those three attacks. Metrics of one kind of attack plays an important role to detect another type; therefore we can expect much more reliable results using our global algorithm compared to the algorithms which consider only the metrics of one attack.

3.4 Metrics

In this section we introduce the metrics that we will use to design our detection model. These are the features that we want to extract from the traffic or with resource usage monitoring. We used the previous works to find which features are useful for us. The references related to each metric will be cited in the related section for explain each metric.

The metrics are selected from prior works on Http attacks such as [54, 81-85], on TCP SYN Flood attacks such as [2, 54, 82, 86] and in Database attacks such as [54, 81, 82].

The selected metrics are as follow:

TSP: Time Spent on a Web Page.

IOWi: Network I/O in Webserver virtual machine (incoming) (Kbits/s).

IOWo: Network I/O in Webserver virtual machine (outgoing) (Kbits/s).

IODi: Network I/O in Database virtual machine (incoming) (Kbits/s).

IODo: Network I/O in Database virtual machine (outgoing) (Kbits/s).

CPUW: Percent of CPU usage in Webserver virtual machine.

CPUD: Percent of CPU usage in Database virtual machine.

MemW: Percent of Memory usage in Webserver virtual machine.

NBWph: Network bandwidth usage in Webserver per hours (MB).

NBWpd: Network bandwidth usage in Webserver per day (MB).

NBWpw: Network bandwidth usage in Webserver per week (GB).

NBWpm: Network bandwidth usage in Webserver per month (GB).

R(SYN): The ratio of SYN packets in TCP packets.

R(ACK): The ratio of ACK packets in TCP packets.

R(SYN+ACK): The ratio of SYN and ACK packets in TCP packets.

NPi: Number of packet (incoming) per second.

NPo: Number of packet (outgoing) per second.

NHOP: Number of half opened connection.

In the following subsections we explain how these metrics can help us to detect attacks. It is important to notice that our goal is not to compare different types of attacks. Rather than that, by comparing the ratio of these metrics we will detect the type of attack. Therefore, we need to understand how these metrics vary in each of the three types of attack.

3.4.1 Time Spent on a Web Page (TSP)

The Time Spent on a Web Page (TSP), which has been previously presented by Koduru et al.[81], will be used as a metric of our work. The TSP is calculated using the following equation [81].

$$\mathrm{TSP}(P_{ij}) = T_{ij+1} - T_{ij} \quad (1)$$

Where P_{ij} represents the accessed page and T_{ij} represents the time stamp of the request for j^{th} page of i^{th} user. Also T_{ij} and T_{ij+1} are two consecutive requests for the webpage.

Amount of TSP in case of HTTP attack suddenly decrease because the attackers send a flood of HTTP requests to the server. On the other hand, when a Database EDOS attack occurs there are a lot of requests sent to database virtual machine while the webserver virtual machine isn't very busy; therefore TSP in this case has a jump. Also since TCP SYN Flood attack happens before completing the access to the webserver, there is no TSP in case of TCP SYN Flood attack.

3.4.2 Network I/O in webserver virtual machine and Database virtual machine

One of the resources which has been investigated in case of DDoS attacks in the privious works is Network I/O [54, 82]. Our goal in this work is to collect the incoming rate and outgoing rate of network I/O in webserver VM and database VM and to investigate the proportion of these metrics.

The amount of network I/O in webserver will be a little high in case of HTTP attack as attackers send a flood of HTTP requests to the server but this amount is still not very different from a busy normal traffic; however, database virtual machine has a very small network I/O in HTTP attack situation because the target of HTTP attack is webserver; therefore, database is almost unemployed in this case and this is the main difference between HTTP attack traffic and busy normal traffic in terms of network I/O values. In the normal situation however, network I/O in webserver virtual machine is almost equal to network I/O in database virtual machine.

Contrary to HTTP attacks, when a Database EDOS attack occurs there is a lot of requests to database virtual machine while webserver virtual machine isn't very busy; so webserver VM has a very small network I/O and database VM has a slightly high rate of network I/O in database attack situation.

Also since TCP SYN Flood attack happens before completing the access to webserver the rate of network I/O will be small in both webserver and database virtual machines in case of this attack.

We will use *IPTraf* for collecting I/O rates which is a tool for monitoring network statistics in webserver virtual machine and database virtual machine.

3.4.3 Percent of CPU usage in webserver virtual machine and database virtual machine

Based on privous works [54, 82] another resource that is affected in DDoS and EDoS attacks is CPU usage. The trend of CPU usage in webserver VM and database VM is similar to network I/O trend in webserver and database VMs in case of HTTP attack, Data base attack and also normal situation. CPU usage in webserver virtual machine is almost equal to CPU usage in database virtual machine in the normal situation. The percent of CPU usage in webserver will be

a little high in case of HTTP attack as attackers send a flood of HTTP requests to webserver virtual machine and webserver has to do a lot of work to answer these requests. On the other hand, database virtual machine has a very small percent of CPU usage in HTTP attack situation because the target of attack is webserver and therefore, database is almost unemployed in this case.

Database attack has an opposite trend to what observed in HTTP attack. When a Database EDOS attack occurs there are a lot of requests to database virtual machine while webserver virtual machine isn't very busy; thus webserver VM has a very small CPU usage and database VM has a slightly high percent of CPU usage in database attack situation. Also in TCP SYN Flood attack the percent of CPU usage in webserver is very high and is almost 2 or 3 times or even 4 times more than amount of this metric in database virtual machine.

The CPU usage data will be collected according to the proposed method by Shea et al. [54]. There are a lot of tools and commands in Linux to collect CPU usage. We are going to use the *Top* command on the hypervisor side for this task because in this command we can monitor CPU usage per each VM separately.

3.4.4 Memory usage in webserver virtual machine

One other metric that we want to use in this work is memory usage in webserver VM. We chose memory usage as a metrics because other researchers in previous works [54, 82] revealed the influence of DDoS attacks on Memory usage. Memory usage in HTTP attack and database attack is very similar to normal situation; although it can grow up very fast in case of TCP SYN Flood attack because this attack directly influences the Memory usage. Thus we have a sudden increase in Memory usage in TCP SYN Flood attack.

We will collect memory usage of each VM by using the *Top* command similar to the last section.

3.4.5 Network bandwidth in webserver

Network bandwidth is one of the most important metrics that plays a key role in our detecting model in case of detecting Bandwidth – consuming attacks like HTTP EDoS attack. Based on previous works [54, 82-85] Network Bandwidth is an important resource which is affected in a DDoS or EDoS attacks.

We expect that bandwidth usage won't change significantly compared to a normal situation neither in TCP SYN Flood attack nor in Database attack. However, bandwidth usage of webserver goes through a sudden increasing in case of HTTP attack.

We will capture the bandwidth usage in webserver virtual machine on the hypervisor side by using *vnstat* per hour, day, week and month.

3.4.6 Packet information

Depending on the type of the attack, a sudden change in a number of network packets can occur in the time of attack. This change may also happen in a heavy perfectly normal traffic. However in a normal traffic the ratio of different types of packets is almost identical, no matter the traffic is heavy or light. But this ratio will change considerably in case of attack traffic. For example, if an attacker uses only the TCP Flood attack in a DDOS attack, the ratios of TCP ACK , TCP SYN and TCP (SYN+ACK) packets are not the same as these ratios in the normal situation.

Therefore, other metrics are as follow:

R(SYN): The ratio of SYN packets in TCP packets.

R(ACK):: The ratio of ACK packets in TCP packets.

R(SYN+ACK):: The ratio of (SYN + ACK) packets in TCP packets.

We will use LTTng [87] for sampling and capturing information from traffics. The LTTng tracer (Linux Tracing Toolkit Next Generation) was expanded by DORSAL lab of Ecole Polytechnique de Montreal University.

3.4.7 Number of packets per second (incoming and outgoing)

The number of packets per second is another metric that will be collected in 2 parts, incoming and outgoing rate, in webserver virtual machine by using *IPTraf* tool similar to collecting I/O rate. As other researcher discused previously[86] the number of packet per second is a very common metric which sudenly change in case of DDoS and EDoS attacks.

The number of packets in webserver will be suddenly increasing in HTTP attack but it is still not very different from a busy normal traffic. This amount will decrease in case of database attack but it is still not very different from a least busy normal traffic. The main difference in number of

packets is in TCP SYN Flood attack compared to a normal traffic. The number of packet in TCP SYN Flood attack suddenly increases significantly.

3.4.8 Number of half-opened connections

Number of half-opened connections is a very important metric that we will use specially for detecting TCP SYN Flood attack by using *netstat* command in Linux.

We expect that the number of half- opened connections in HTTP attack and database attack will be almost the same as the normal situation. But this parameter will suddenly increase in case of TCP SYN Flood attack [2]. In TCP SYN Flood, attackers send a huge number of SYN packets to victim and also they don't answer to ACK packet which is sent by the server (victim); so the three-way handshake cannot be completed and connection is released without completion. Thus we will have a huge number of half –opened connections in which SYN+ACK packet is lost.

3.5 Change Point Detection algorithm: CUSUM

For analyzing time ordered data to determine whether a change has taken place a change-point analysis can be performed using CUSUM algorithm [88, 89]. In our project we have used a procedure introduced by Taylor[89] in which a combination of cumulative sum charts (CUSUM) and bootstrapping is used to detect the changes.

The first step in this analysis is to create the CUSUM chart based on a cumulative sum of the data. For n data points represented by Y_1, Y_2, …, Y_n (Y_i represents the data point i) the cumulative sums S_0, S_1, …, S_n are calculated from the average of data. The cumulative sum starts at 0 by setting S0 = 0. Then we calculate the other cumulative sums by adding the difference between current value and the average to the previous sum as follows:

$$S_i = S_{i-1} + (Y_i - \overline{Y}) \qquad (2)$$

for

i = 1, 2, …, n.

Where $\bar{Y}$ is the average of n data:

$$\overline{Y} = \frac{\sum_{i=1}^{n} Y_i}{n} \qquad (3)$$

The cumulative sums calculated in this method are different from the normal cumulative sums of the values. In this method we consider the cumulative sums of variances between the values and the average. This cumulative sum always terminates at zero (S_n=0) because variances between the values and the average sum to zero.

By plotting the CUSUM chart we can get some ideas as to whether or not a change has been occurred. When the CUSUM chart has an upward slope the values lean toward above the overall average. A downward slope is related to a period of time where the values lean toward below the overall average. A sudden shift or change in the average can be detected by a sudden change in direction of the CUSUM. If the average is constant CUSUM chart follows a relatively straight path.

To be rest assured about the changes a bootstrap analysis can be performed to determine a confidence level for the apparent changes. But first we need "an estimator of the magnitude of the change". S_{diff} which is the difference between S_{max} and S_{min} is a good choice in this case as it works well for any distribution and multiple changes. S_{diff} is calculated as:

$$S_{diff} = S_{max} - S_{min} \quad \text{where:}$$

$$\begin{cases} S_{max} = \max S_i \ \ (\mathrm{i} = 0, \dots \mathrm{n}) \\ S_{min} = \min S_i \ \ (\mathrm{i} = 0, \dots \mathrm{n}) \end{cases} \qquad (4)$$

To perform a single bootstrap the procedure is as follow:

1. First the original *n* values are randomly reordered to create a bootstrap sample of n units, denoted as $Y^0_1, Y^0_2, \dots, Y^0_n$,.
2. The bootstrap CUSUM will be calculated for the new bootstrap sample as $S^0_0, S^0_1, \dots, S^0_n$
3. The maximum, minimum and difference of the bootstrap CUSUM will be calculated as S^0_{max}, S^0_{min}, and S^0_{diff} respectively.
4. The bootstrap difference will be compared with the original difference to define if the bootstrap difference S^0_{diff} is less than the original difference S_{diff}.

Bootstrapping analysis is based on the fact that if no change has taken place bootstrap samples (which represent random reorderings of the data) would copy the behavior of the CUSUM.

Accordingly, a large number of bootstraps will be performed and the number of bootstraps for which S^0_{diff} is less than S_{diff} will be counted. The confidence level (as a percentage) of a change occurrence can be calculated as follows:

$$\text{Confidence Level} = 100\,\frac{P}{M}\,\% \qquad (5)$$

Where M is the number of performed bootstraps and P is the number of bootstraps for which $S^0_{diff} < S_{diff}$.

Normally if the confidence level is about 90%, or 95% we can conclude the detection of a significant change. For n samples the total number of possible reorderings is n!. In a bootstrap analysis we estimate the distribution of S^0_{diff} by randomly selecting 1000 of these possible reorderings. It has been proven that 1000 bootstraps is ample in most cases.

After detecting a change, the CUSUM estimator can be used to estimate when the change happened. If the change occurred at k , then:

$$|S_k| = \max |S_i| \text{ for } i=1...n \qquad (6)$$

S_k is the furthest point from zero in the CUSUM chart. k is the last point before the change and k+1 is the first point after the change.

To detect multiple changes, after detecting the first change, the data is split into two parts, in two sides of the change-point, and each part is analyzed separately. The splitting procedure will continue for any additional significant change.

In order to identify whether the detected change point is more than normal situation or less than the normal situation we use α as the slope of change point chart between k (change point) and k-1 (the point before change point). α can be calculated as follow:

$$\alpha = \frac{S_k - S_{k-1}}{Y_k - Y_{k-1}} \qquad (7)$$

In equation7, $\alpha > 0$ means the chart of CUSUM has an upward slope and indicates that k is a change point with a value less than normal situation (sudden decrease). Likewise $\alpha < 0$ indicates a downward slope in CUSUM chart and therefore k is a change pointe with a value more than normal situation (sudden increase).

3.6 Proposed algorithm to detect different types of DDoS and EDoS attacks

In this section, we propose an algorithm to detect different types of attacks (HTTP attack, Database attack and TCP SYN Flood attack) by using those metrics that we named in the last section. This algorithm works based on a state parameter, addressed as S_{tp} in the following subtitles, where t is the type of attack (Http attack, Database attack or TCP SYN Flood) and ***p*** represents the sign ***p*** in each attack. The state of each attack and the detail of our algorithm will be explained in the following sections for Http attack, Database attack and SYN Flood attack situation. Later on we combine all the states of three types of attack and check all of them using their state parameters in our algorithm to provide the probability of each attack. But first we need to understand how we can determine the value of state parameters in each attack.

3.6.1 Http attack sings

When a virtual machine applies for more resources and sends its requests to the hypervisor, hypervisor must analyze the VM which sent requests and VMs which communicated with that VM and must then check the bellow states. If hypervisor finds the bellow states in the behavior of VMs, we have an Http attack in that VM and hypervisor should not allocate more resources to that VM.

The Http attack signs are as follow:

- Sign 1: TSP in Http attacks shows a sudden decrease.
- Sign 2: IOWi in Http attacks is very larger than IODI.
- Sign 3: IOWo in Http attacks is very larger than IODO.
- Sign 4: CPUW in Http attacks is very larger than CPUD.
- Sign 5: NBWph in Http attacks shows a sudden increase.
- Sign 6: NBWpd in Http attacks shows a sudden increase.
- Sign 7: NBWpw in Http attacks shows a sudden increase.
- Sign 8: NBWpm in Http attacks shows a sudden increase.

The State parameter for the sign p in this attack is defined as $S_{Http, p}$. For the sign p in Http attack, $S_{Http, p}$ is equal to 1 if the sign meets and is equal to 0 if the sign doesn't meet.

3.6.2 Database attack signs

Similar to Http attack, when a virtual machine applies for more resources and sends its requests to the hypervisor, hypervisor must inspect the VM which sent requests and VMs which communicated with that VM and must then check the bellow states. If hypervisor finds the bellow states in the behavior of VMs, we have a Database attack in that VM and hypervisor should not allocate more resources to that VM. The states that specify a Database attack is almost the opposite of the Http attack's states.

The Database attack signs are as follow:

- Sign 1: TSP in Database attacks shows a sudden increase.
- Sign 2: IOWi in Database attacks is very smaller than IODI.
- Sign 3: IOWo in Database attacks is very smaller than IODO.
- Sign 4: CPUW in Database attacks is very smaller than CPUD.

The State parameter for the sign p in this attack is defined as $S_{Database, p}$. For the sign p in Database attack, $S_{Database, p}$ is equal to 1 if the sign meets and is equal to 0 if the sign doesn't meet.

3.6.3 TCP SYN Flood attack signs

Similar to last two sections, there are some states that should be checked by hypervisor in case of TCP SYN Flood attacks. Some of the bellow metrics are retrieved from the literatures and some of them have been identified during our testing and network traffic analyzing as will be explained in details in chapter (4).

The TCP SYN Flooding attack signs are as follow:

- Sign 1: MemW in TCP SYN Flood attacks shows a sudden increase.
- Sign 2: In our testing and samples analyzing we understood that R(SYN), R(ACK) and R(SYN+ACK) are somehow related to each other in case of SYN Flood attacks and

normal situation as we have shown in table 1; where T. specifies that the relation is valid and F. indicates that the relation is not valid.

- Sign 3: NPi in TCP SYN Flood attacks shows a sudden increase.
- Sign 4: NPo in TCP SYN Flood attacks shows a sudden increase.
- Sign 5:NHOP in TCP SYN Flood attacks shows a sudden increase.

The State parameters in this attack will be defined according to the following procedure:

- The state parameter of Sign 1 in TCP SYN Flooding attack, $S_{\text{TCP SYN Flooding, 1}}$ is equal to 1 if the sign meets and is equal to 0 if the sign doesn't meet.
- The state parameter of Sign 2 in TCP SYN Flooding attack, $S_{\text{TCP SYN Flooding, 2}}$ is defined as shown in table 3.1.
- The state parameter of Sign 3 in TCP SYN Flooding attack, $S_{\text{TCP SYN Flooding, 3}}$ is equal to 1 if the sign meets and is equal to 0 if the sign doesn't meet.
- The state parameter of sign 4 in TCP SYN Flooding attack, $S_{\text{TCP SYN Flooding, 4}}$ is equal to 1 if the sign meets and is equal to 0 if the sign doesn't meet.
- The state parameter of sign 5 in TCP SYN Flooding attack, $S_{\text{TCP SYN Flooding, 5}}$ is equal to 1 if the sign meets and is equal to 0 if the sign doesn't meet.

Table 3. 1:Relationship between R(SYN), R(ACK) and R(SYN+ACK)

R(SYN)= 2R(SYN+ACK)	R(ACK)+R(SYN+ACK)= Total number of samples' packet	R(SYN)+ R(SYN+ACK) >=Total number of samples' packet	R(ACK)= R(SYN+ACK)	Traffic Situation	State Parameter
T.	T.	F.	F.	High chance of Normal	-3
T.	T.	F.	T.	Medium chance of Normal	-2
T.	T.	T.	F.	Medium chance of Normal	-2
T.	F.	F.	F.	Low chance of Normal	-1
F.	T.	F.	F.	Low chance of Normal	-1
F.	F.	T.	T.	High chance of Attack	3
F.	T.	T.	T.	Medium chance of Attack	2
T.	F.	T.	T.	Medium chance of Attack	2
F.	F.	F.	T.	Low chance of Attack	1

Table 3. 1:Relationship between R(SYN), R(ACK) and R(SYN+ACK)(cont'd)

F.	F.	T.	F.	Low chance of Attack	1
T.	T.	T.	T.	Unknown	0
F.	F.	F.	F.	Unknown	0
T.	F.	T.	F.	Unknown	0
F.	T.	T.	F.	Unknown	0
T.	F.	F.	T.	Unknown	0
T.	F.	F.	T.	Unknown	0

Table 3.1 shows the relationship between R(SYN), R(ACK), R(SYN+ACK). The first two columns (columns 1,2) represent this relationship in an ideal normal situation and the next two columns (columns 3,4) are representing this relationship in an ideal attack situation. A negative state parameter means that this parameter shifts the traffic to a normal one and therefore it decreases the probability of the attack.

3.6.4 Detecting changes by using CUSUM algorithm:

In this work, we will use CUSUM to detect changes in TSP, MemW, NBWph, NBWpd, NBWpw, NBWpm, NPi, NPo, NHOP metrics. Then the signs that we explained in sections 3.6.1, 3.6.2 and 3.6.3 will be considered to detect traffic situation.

In this thesis "*CUSUM_ Implementation"* is a function that we employed to implement CUSUM. As you see in the bellow code, if $\alpha > 0$, this function returns value 0 and indicates that k is a change point with a value less than normal situation (sudden decrease) and if $\alpha < 0$, it returns value 1 that means k is a change pointe with a value more than normal situation (sudden increase).

CUSUM_Implementation :
$\overline{Y}$ *= average of* Y_i
$S_0=0$
For i=1, i<= n, i++:
$S_i = S_{i-1} + (Y_i - \overline{Y})$
End For
$S_{diff} = S_{Max} - S_{min}$
For j=1, j<= 1000 , j++:
Y = boot strap of Y
$S^0_0=0$
For i=1, i<= n, i++:
$S^0_i = S^0_{i-1} + (Y_i - \overline{Y})$
End For
$S^0_{diff} = S^0_{Max} - S^0_{Min}$
If $S^0_{diff} < S_{diff}$ *then:*
$m=m+1$
End If
End For
If $\frac{m}{1000} * 100 >= 90$ *then:*
$|S_k| = Max\ |S_i|$
$\alpha = \frac{S_k - S_{k-1}}{Y_k - Y_{k-1}}$
If $\alpha <= 0$ *then:*
Return 1
Else:
Return 0
End If
End If
Print "there is no change point"
End CUSUM_Implementation

Then we give the values of **TSP** as input to CUSUM_Implementation to find and to detect any changes in **TSP** values. Thus we will have:

$$Y \leftarrow TSP$$

So if the function returns value 0, it indicates that k is a change point with a **TSP** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with a **TSP** value more than normal situation (sudden increase). Otherwise there is no significant change in **TSP** values.

Afterward we will consider sign 1 in HTTP Attack, Database attack and will calculate $S_{\text{HTTP attack}, 1}$ and $S_{\text{Database attack}, 1}$.

In the next step we give the values of **MemW** as input to CUSUM_Implementation to find and to detect any changes in **MemW** values. Thus we will have:

$$Y \leftarrow \text{MemW}$$

So if the function returns value 0, it indicates that k is a change point with a **MemW** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with a **MemW** value more than normal situation (sudden increase). Otherwise there is no significant change in **MemW** values.

Subsequently we will consider sign 1 in TCP SYN Flood attack and will calculate $S_{\text{TCP SYN Flood}, 1}$.

Also we give the values of **NBWph, NBWpd, NBWpw, NBWpm** as input to CUSUM_Implementation to find and to detect any changes in **NBWph, NBWpd, NBWpw, NBWpm** values separately. We will use CUSUM_Implementation for each of these 4 metrics (NBWph, NBWpd, NBWpw, NBWpm) separately similar to what previously mentioned for TSP and MemW. Therefore we have to run the function for 4 more times. For the first time we will have:

$$Y \leftarrow \text{NBWph}$$

For the second time of running the function we will have:

$$Y \leftarrow \text{NBWpd}$$

For the third time we will have:

$$Y \leftarrow \text{NBWpw}$$

And finally for the fourth time we will have:

$$Y \leftarrow \text{NBWpm}$$

In the first time if the function returns value 0, it indicates that k is a change point with a **NBWph** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with a **NBWph** value more than normal situation (sudden increase). Otherwise there is no significant change in **NBWph** values. Afterward we will consider sign 5 in HTTP attack and will calculate $S_{\text{HTTP attack},5}$.

In the second time if the function returns value 0, it indicates that k is a change point with a **NBWpd** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with a **NBWpd** value more than normal situation (sudden increase). Otherwise there is no significant change in **NBWpd** values. Afterward we will consider sign 6 in HTTP attack and will calculate $S_{\text{HTTP attack},6}$.

In the third time that we will implement the function, if it returns value 0, it shows the k is a change point with a **NBWpw** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with a **NBWpw** value more than normal situation (sudden increase). Otherwise there is no significant change in **NBWpw** values. Afterward we will consider sign 7 in HTTP attack and will calculate $S_{\text{HTTP attack},7}$.

Finally in the fourth time if the function returns value 0, it indicates that k is a change point with a **NBWpm** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with a **NBWpm** value more than normal situation (sudden increase). Otherwise there is no significant change in **NBWpm** values. Afterward we will consider sign 8 in HTTP attack and will calculate $S_{\text{HTTP attack},8}$.

In the next step we give the values of **NPi** and **NPo** as input to CUSUM_Implementation to find and to detect any changes in **NPi** and **NPo** values separately. We will run CUSUM_Implementation 2 times for 2 above metrics (NPi and NPo). Thus for the first time we will have:

$$Y \leftarrow \text{NPi}$$

And for the second time we will have:

$$Y \leftarrow \mathrm{NPo}$$

In the first time if the function returns value 0, it indicates that k is a change point with an **NPi** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with an **NPi** value more than normal situation (sudden increase). Otherwise there is no significant change in **NPi** values.

Afterward we will consider sign 3 in TCP SYN Flood attack and will calculate $S_{\text{TCP SYN Flood},3}$.

In the second time if the function returns value 0, it indicates that k is a change point with an **NPo** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with an **NPo** value more than normal situation (sudden increase). Otherwise there is no significant change in **NPo** values. Afterward we will consider sign 4 in TCP SYN Flood attack and will calculate $S_{\text{TCP SYN Flood},4}$.

Finally we give the values of **NHOP** as input to CUSUM_Implementation to find and to detect any changes in **NHOP** values. Thus we will have:

$$Y \leftarrow \mathrm{NHOP}$$

So if the function returns value 0, it indicates that k is a change point with a **NHOP** value less than normal situation (sudden decrease) and if the function returns value 1 that means k is a change pointe with a **NHOP** value more than normal situation (sudden increase). Otherwise there is no significant change in **NHOP** values. Afterward we will consider sign 5 in TCP SYN Flood attack and will calculate $S_{\text{TCP SYN Flood},5}$.

3.6.5 Detecting attack percentage procedure

As mentioned earlier, one of our contributions in this work is to detect different types of attack in one algorithm. To achieve this goal we need to combine the states of all three types of attack to provide a probability of each attack. The percentage of each attack is defined as follow:

$$W_t[\%] = \frac{\sum_{p=1}^{n} Stp}{\sum_{p=1}^{n} Smax\ tp} * 100 \quad (8)$$

W_t is the percentage of the attack type t. For the attack t with n signs, W_t is defined as the ratio of the sum of the real values of all n state parameters to the sum of the maximum values that n state parameters can have.

3.7 Conclusion

In this chapter we proposed a novel and inclusive model for detecting different types of EDoS and DDoS attacks. In designing our model, we used the metrics related to all 3 types of attacks and consequently we could successfully introduce a global algorithm based on the metrics that previously introduced in previous works. This model works perfectly for all three of HTTP attack, Database attack and TCP SYN Flood attack no matter what time each attack happened. Therefore, instead of using three different algorithms for detecting these attacks, we are able to use a generalized algorithm. Furthermore, this precise detection is considered as a defense itself because the services will be provided only to VMs that are in Normal situation; therefore we prevent the migration of attacks from VM to the hypervisor. This is a temporary defense for the system till finding a permanent solution in the future. Moreover, we calculated an attack persentage value for each type of attack .The critical value of the attack percentage to make a decision depends on the sensitivity of the system. For more sensitive system, even attacks with low percentage causes limiting resources.

In the next chapter we present our experiments and our results. Furthermore, we will compare our detection rate with privous works and will also evaluate our metrics in chapter 4.

CHAPTER 4 RESULTS AND DISCUTIONS

As discussed earlier, the virtualization is the main concept of cloud computing and every service in cloud computing is located in a virtual machine. For example, the database is placed in one virtual machine and the web server is located in another virtual machine. Therefore when customers want to put their services in the cloud, they must order a set of VMs (we consider the infrastructure as a service). However, cloud computing system is a pay-as-you-go service, so customers have to pay for the amount of VM that they used. They can apply for more resources whenever they want.

On the hypervisor side we can monitor all the events and the requests from each VM. In the proposed model, when a virtual machine applies for more resources and sends its requests to the hypervisor we can have a monitoring system to trace the VM kernel system and analyze the traffic. If the situation is normal, the resources are allocated to VM, but if an attack is detected in VM resources, the resources will not be allocated to VM and the live migration will be stopped.

Based on this explanation, we allocate one VM to the web server and another VM to database server and try to extract metrics that have been explained in section 3.3. Finally, metrics of the new arrival traffic will be given to our model as input and the proposed model will detect the situation of this traffic. We are going to clarify in this chapter how to extract metrics of our model. At the end of this chapter we will discuss our detection results and compare our model with previous works in order to evaluate the performance of our proposed model.

4.1 Set up for experimental result

In this section we are going to explain how we did the experimental result where we hypothesize that the attacker is outside of the cloud. Another hypothesis in this scenario is that we know server of a big company is located in the Virtual machine in a cloud computing environment. At the first glance it may look like that the attack is similar to a traditional attack (attack in non-virtualization environment); but in fact when a server is in VM, resources of this server are extendable and increasable.

We used Qemu [90] (version 2.0.0-rc1) that is a common and open source machine emulator and virtualization machine to have VMs base on KVM. We considered one of the VMs as a webserver virtual machine and run a webserver on it. We used Httpd[91] version 2.2 that is

Apache HyperText Transfer Protocol program for server. Also we considered one other VM as a Database server (these two VMs (webserver and Database server) are as a server of a company). MySQL[92] version 5.6.16 that is the world's most popular open source database was used in this VM.

Table 4.1 can show the detail of physical machine and VMs configurations that used in this work.

4.1: System details

	Memory	CPU
Physical Machine	7.7 GiB	Intel core2 Quad CPU Q6700 @2.66 GHz × 4
Virtual Machines	2.0 GiB	QEMU Virtual CPU version 1.0 × 2

In the next step we consider some customers for this company and generate some normal traffics from customers to server by using netsniff-ng toolkit [93] that can generate a normal traffic. We used *trafgen* command to generate normal packets and configured it to send 20000 packets as normal traffic for first time. We changed the amount of packets in different test times.

In the next Phase we implemented TCP SYN Flood attack by using hping3 tool [94, 95]. We used *hping3 -c* command to generate TCP SYN Flood packets and configured it to send 1000000 packets as TCP SYN Flood traffic. We increase this amount of packets for performing more power attack in next test times.

In the next steps we performed Http attack and Data base attack separately by using HTTPFlooder [96] and LoadRunner [97], consecutively. It is possible to create Http attack by using *httpflooder* command and use appropriate key for generate arbitrary attacks (for example GF is a key for generating Get Flood attacks and PF is a key for generating Post Flood attacks).

In the next step we configured LoadRunner tool for generate more load to database server and assumed it as a Database attack.

Afterwards, we needed some tools for capturing the sample from traffics that generated in the last steps and for monitoring resources too.

We measured Network I/O and the number of packets per second by using *IPTraf* which is a tool for monitoring network statistics. Also *vnstat* is used for measuring the different amount of bandwidth usage. Memory usage and CPU usage is monitored by using *Top* command. The command that we used for monitoring the number of half open-conections was *netstat*. Moreover, We used LTTng (Linux Tracing Toolkit Next Generation) [87] for sampling and capturing Packet information from traffics.

In continue, we explain how we got the samples and analyse the information from traffic and resourse usage in more details.

4.2 Time Spent on a Web Page (TSP)

We used Time Spent on a Web Page (TSP) which has been previously presented by Koduru et al.[81]. The TSP is calculated by using equation (1). On the other hand any sudden changes in the calculated TSP are detected by using CUSUM method. Figures 4.1 and 4.2 show the values of TSP in different numbers of test and the results of implementing CUSUM for TSP in Http attack and database attack respectively.

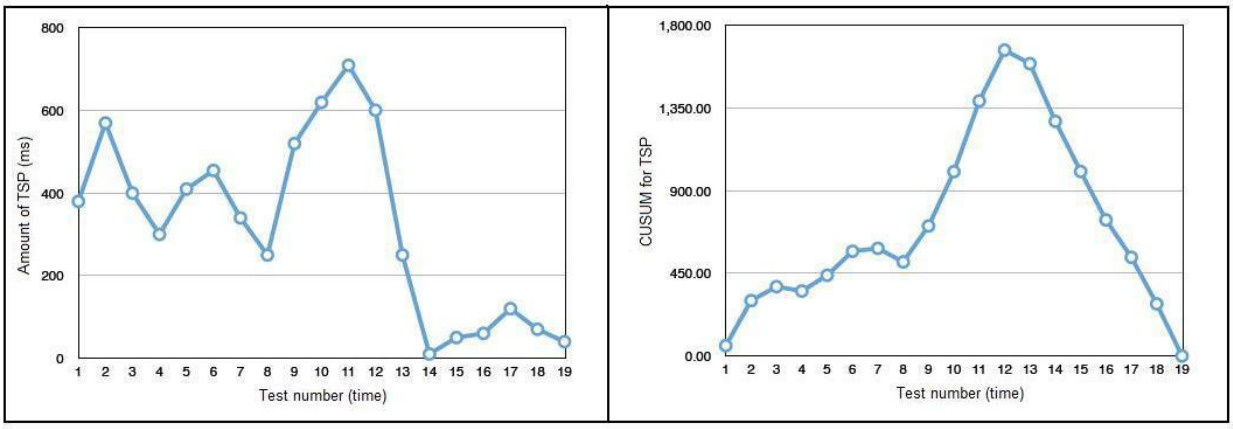


Figure 4.1: variation of TSP values (left) and CUSUM values in TSP(right) in Http attack

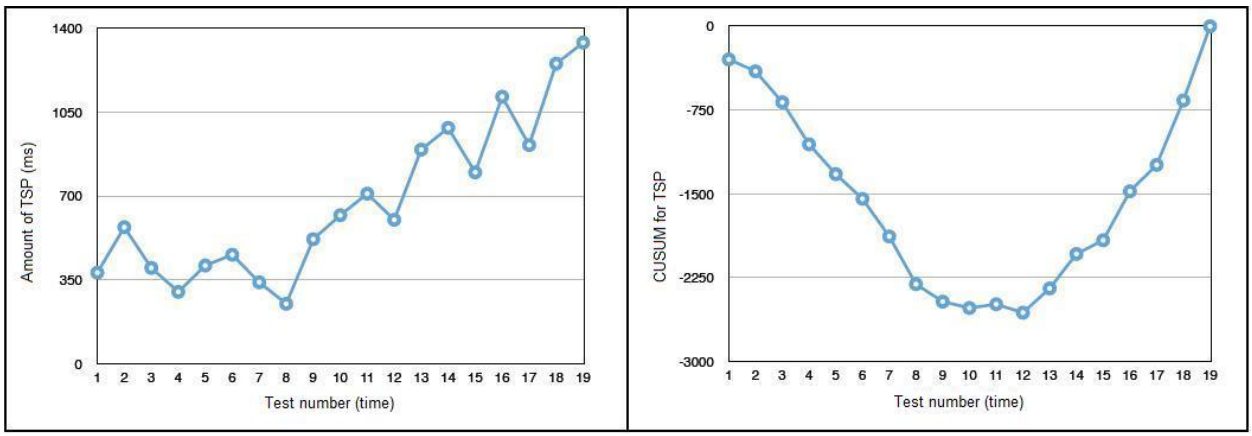


Figure 4.2: variation of TSP values (left) and CUSUM values in TSP(right) in Database attack

As it is clearly shown in the above figures, detecting change points in left pictures is almost impossible, but in right pictures it is very easy to detect change points.

Similarly, there is a change point in figure 4.1 (right) between number of test 12 and 13. Also since the slope of CUSUM chart before this point (number of test 12) is upward so we have a change point with a value less than normal situation in Http attack (sudden decrease).

In case of Database attack, a change has occurred in figure 4.2 between number of test 11 and 13. In this time, there is an downward slope before this point in CUSUM chart for Database attack so we have a change point with a value more than normal situation (sudden increase).

4.3 Network I/O in webserver virtual machine and Database virtual machine

We measured Network I/O in Webserver virtual machine by using *IPTraf* which is a tool for monitoring network statistics. As you see in Figure 4.3, we collected incoming rate and outgoing rate of network I/O in our webserver virtual machine. I/O in database VM is also measured in a similar way by using *IPTraf* tool.

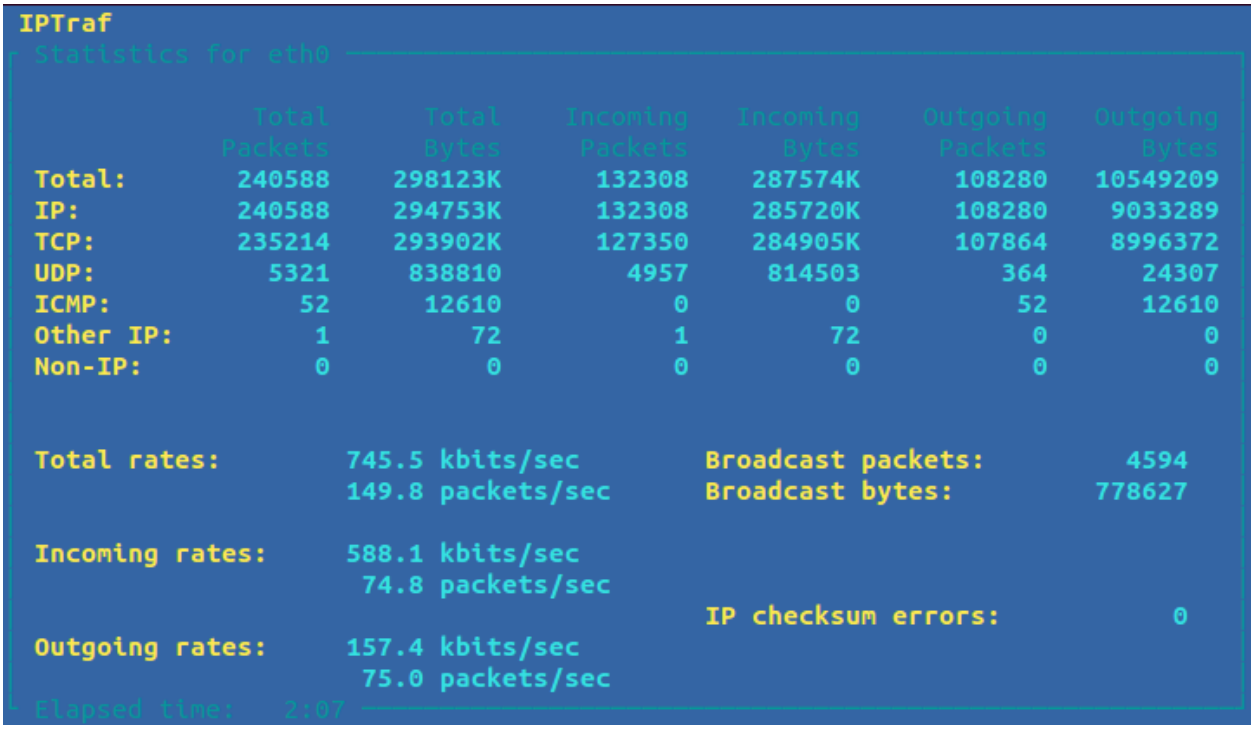


Figure 4.3 : I/O usage in webserver virtual machine in normal situation by using IPTraf

Figure 4.4 shows different amounts of network I/O for webserver virtual machine and database virtual machine in normal, HTTP attack, Database attack and TCP SYN Flood situations.

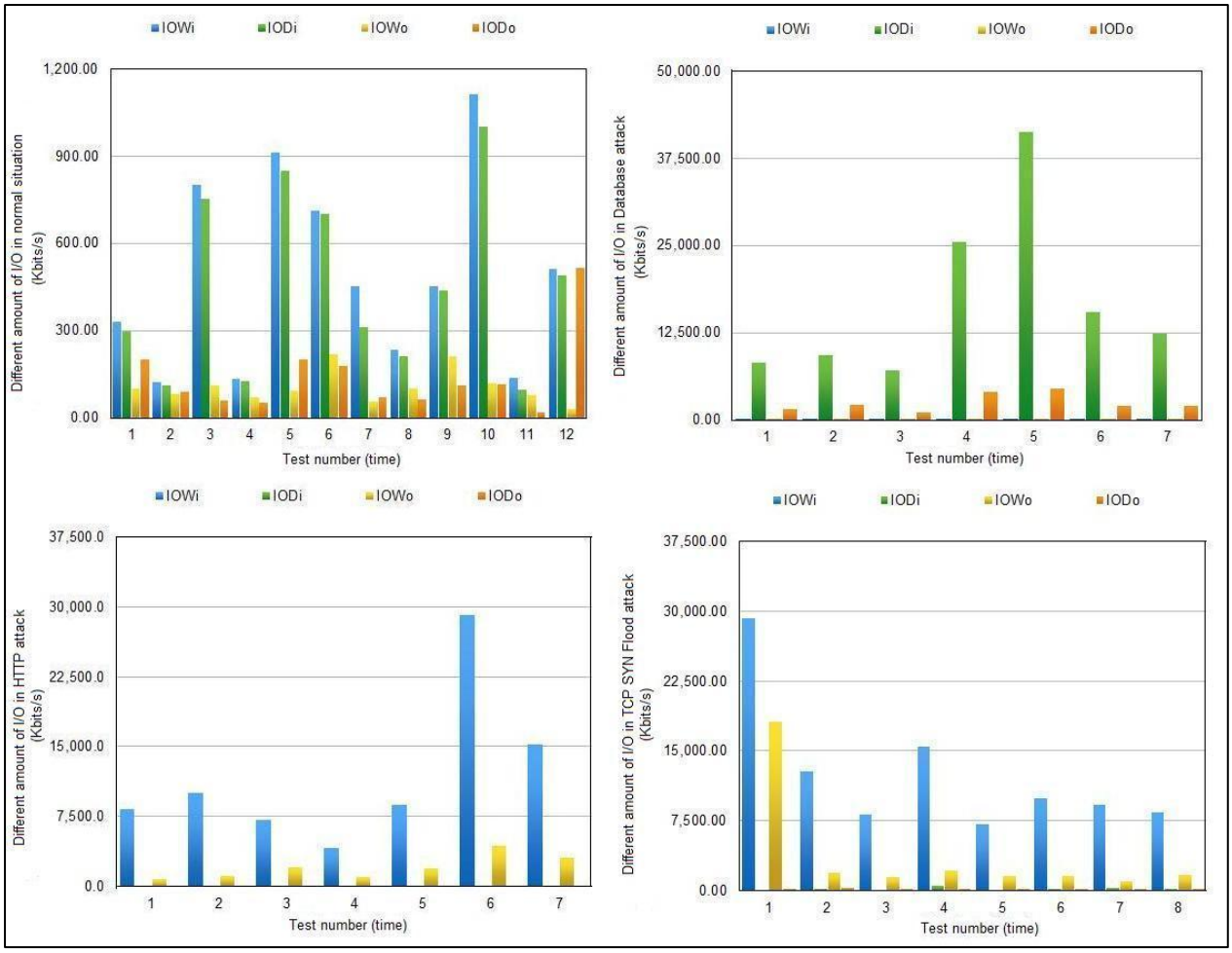
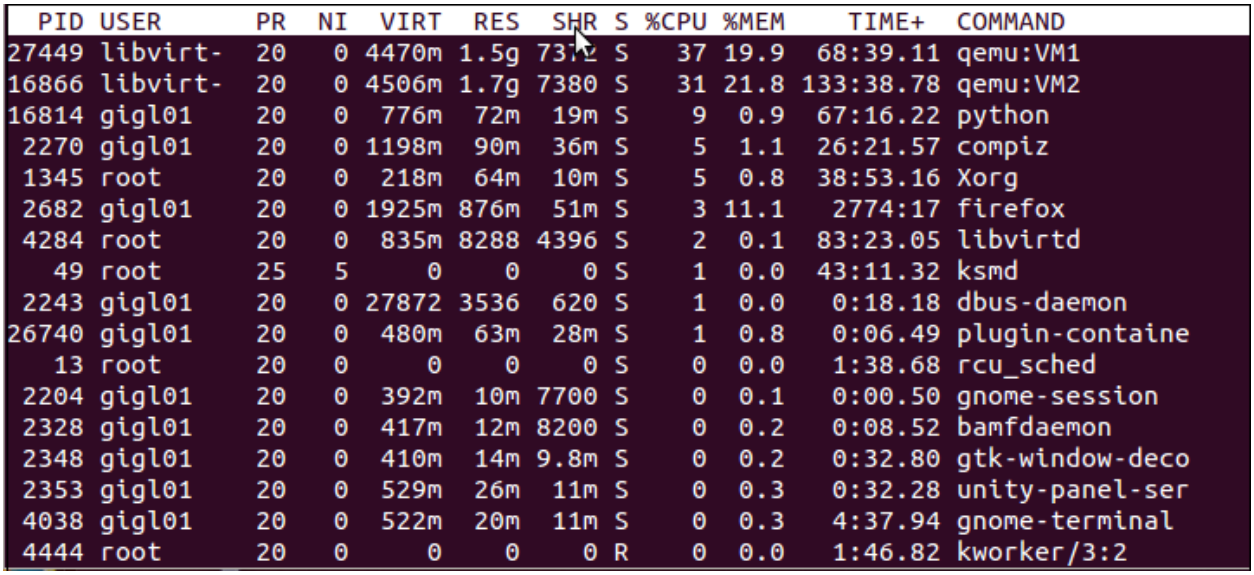


Figure 4.4:Different amount of network I/O in different situation

4.4 Percent CPU usage in webserver virtual machine and database virtual machine

The CPU usage data has been collected according to the proposed method by Shea et al. [54]. There are a lot of tools and commands in Linux to collect CPU usage. We used the Top command on the hypervisor side for this task because in this command we can monitor CPU usage per each VM separately. Figure 4.5 demonstrates our results obtained using the Top command in the hypervisor. CPU usage per each VM (VM1 and VM2) is observed in this figure.



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
27449	libvirt-	20	0	4470m	1.5g	73??	S	37	19.9	68:39.11	qemu:VM1
16866	libvirt-	20	0	4506m	1.7g	7380	S	31	21.8	133:38.78	qemu:VM2
16814	gigl01	20	0	776m	72m	19m	S	9	0.9	67:16.22	python
2270	gigl01	20	0	1198m	90m	36m	S	5	1.1	26:21.57	compiz
1345	root	20	0	218m	64m	10m	S	5	0.8	38:53.16	Xorg
2682	gigl01	20	0	1925m	876m	51m	S	3	11.1	2774:17	firefox
4284	root	20	0	835m	8288	4396	S	2	0.1	83:23.05	libvirtd
49	root	25	5	0	0	0	S	1	0.0	43:11.32	ksmd
2243	gigl01	20	0	27872	3536	620	S	1	0.0	0:18.18	dbus-daemon
26740	gigl01	20	0	480m	63m	28m	S	1	0.8	0:06.49	plugin-containe
13	root	20	0	0	0	0	S	0	0.0	1:38.68	rcu_sched
2204	gigl01	20	0	392m	10m	7700	S	0	0.1	0:00.50	gnome-session
2328	gigl01	20	0	417m	12m	8200	S	0	0.2	0:08.52	bamfdaemon
2348	gigl01	20	0	410m	14m	9.8m	S	0	0.2	0:32.80	gtk-window-deco
2353	gigl01	20	0	529m	26m	11m	S	0	0.3	0:32.28	unity-panel-ser
4038	gigl01	20	0	522m	20m	11m	S	0	0.3	4:37.94	gnome-terminal
4444	root	20	0	0	0	0	R	0	0.0	1:46.82	kworker/3:2

Figure 4.5 : CPU usage

Figure 4.6 shows different amounts of CPU usage in webserver virtual machine and database virtual machine in normal, HTTP attack, Database attack and TCP SYN Flood situations.

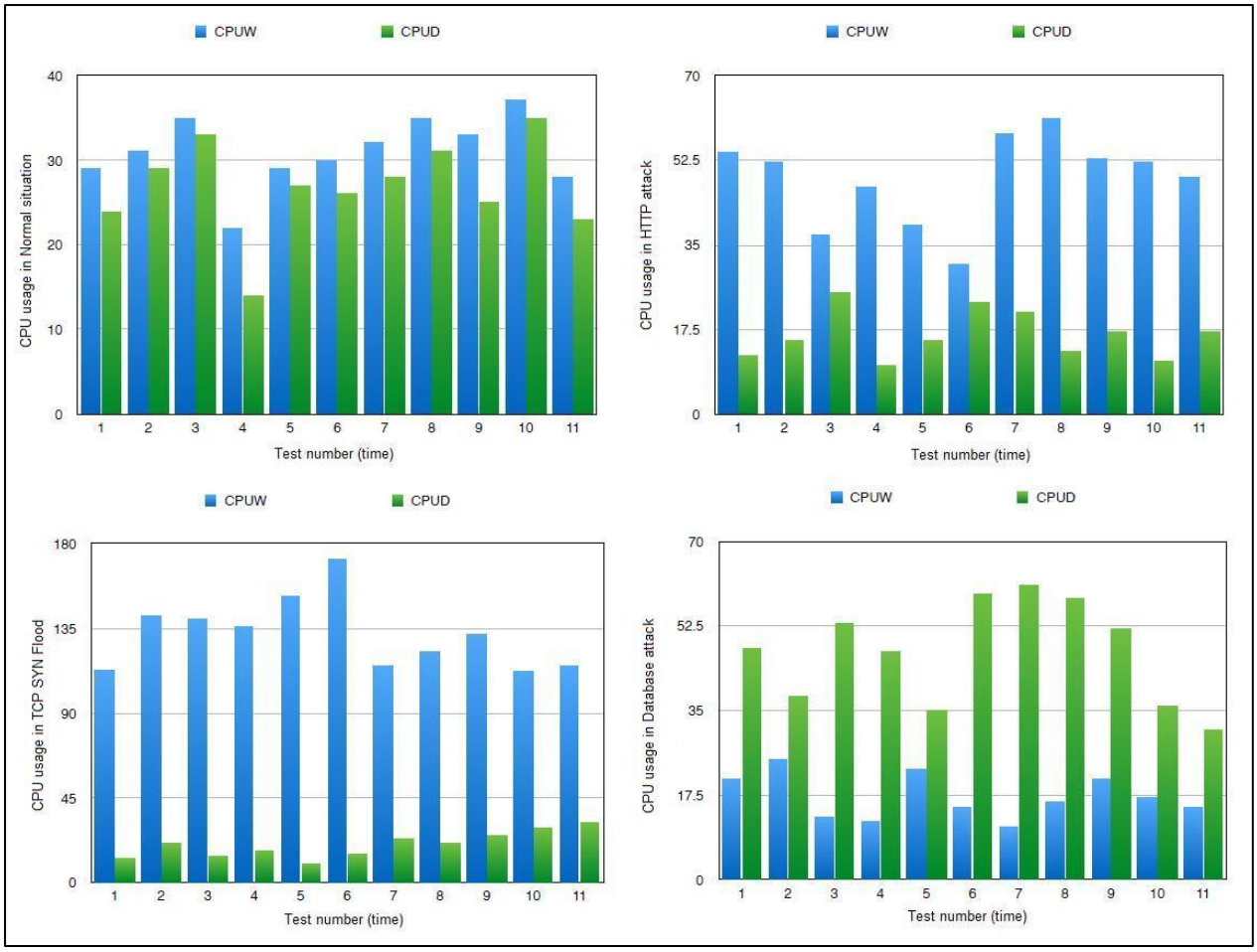


Figure 4.6:CPU usage in different situation

4.5 Memory usage in webserver virtual machine

Memory usage in webserver which is another metric of our work can be collected by using the Top command in the hypervisor similar to CPU usage which had been shown in Figure 4.5. In order to compare this value with normal situation and detect any sudden changes in memory usage, we implement attacks and normal traffic in our system. Memory usage in webserver is compared with normal situation each time by using CUSUM method. Figures 4.7 shows the values of Memory usage (MemW) in different numbers of test and the result of implementing CUSUM for MemW in TCP SYN Flood attack.

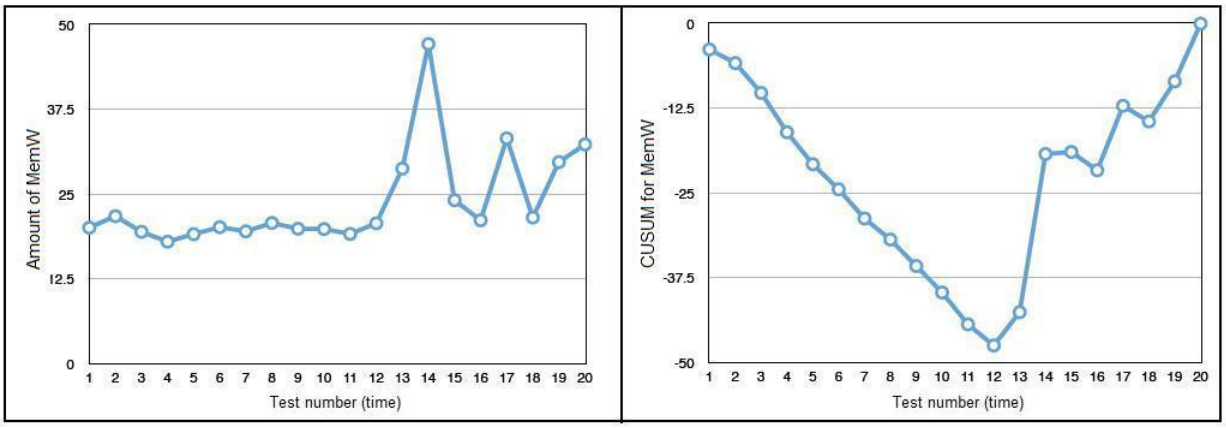


Figure 4.7: variation of MemW values (left) and CUSUM values in MemW (right) in TCP SYN Flood attack

As it is clear in the above figure, detecting change points in the left picture is very difficult, but in the right picture it is very easy to detect the change point. A sudden change happened in figure 4.7 (right) between number of test 12 and 13. Also since the slope of CUSUM chart before this point (number of test 12) is downward so we have a change point with a value more than normal situation in case of TCP SYN Flood attack (sudden increase).

There is no significant change point in Http attack and Data base attack compared to normal situation.

4.6 Network bandwidth in webserver

As mentioned earlier, we ran a web server on a virtual machine and tried to connect to this server from other computers. We captured the bandwidth usage in the webserver virtual machine on the hypervisor side. To have a complete and thorough detection, extensive and comprehensive information was required; therefore, we measured the bandwidth usage per hour, day, week and month by using *vnstat* as you can see in Figure 4.8, Figure 4.9, Figure 4.10 and Figure 4.11 respectively.

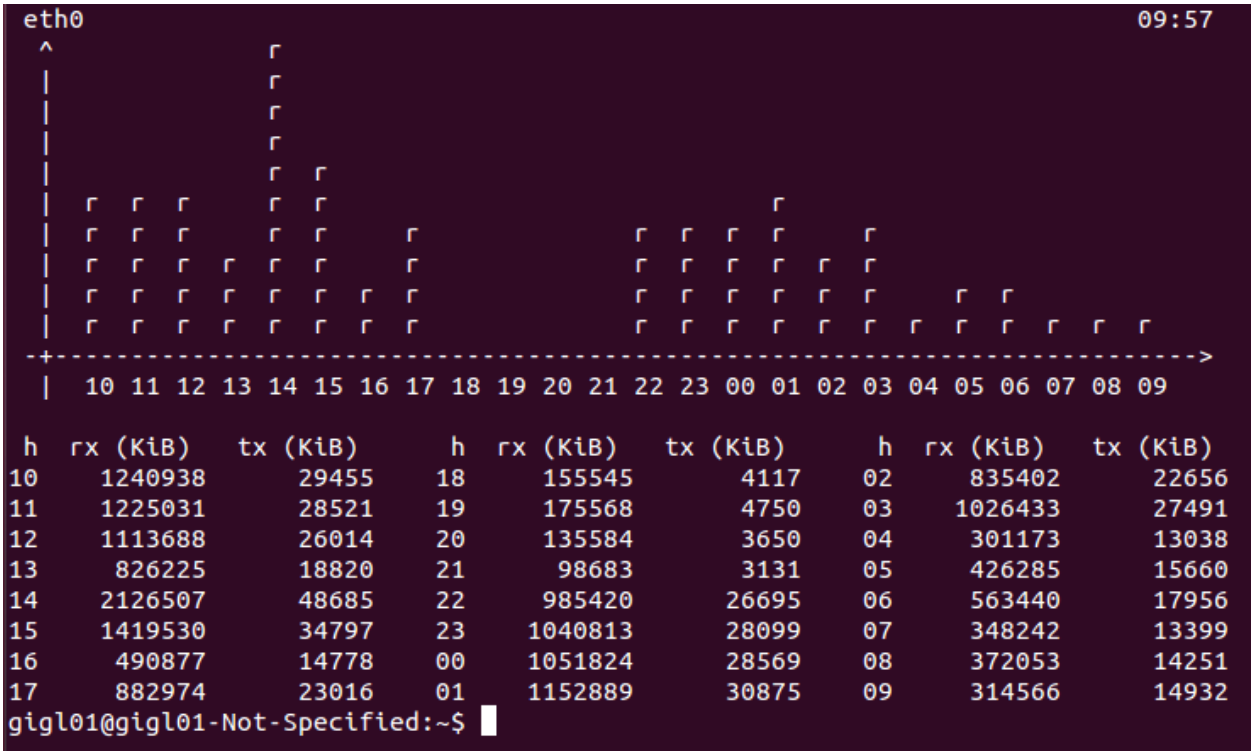


Figure 4.8 : Bandwidth usage per hour

In the table shown in Figure 4.8, *rx* represents the download traffic and *tx* means the upload traffic. As you can see in this figure, the usage of the bandwidth for *rx* is between 98,683 KB and 2,126,507 KB and for *tx* is between 3,131 KB and 48,685 KB during this specific day. Consequently, almost the busiest hour is 2 pm and the least busy hour is 9 pm in one sample day of our web server. Figure 4.8 also shows a diagram demonstrating the ratio of bandwidth usage per hour.

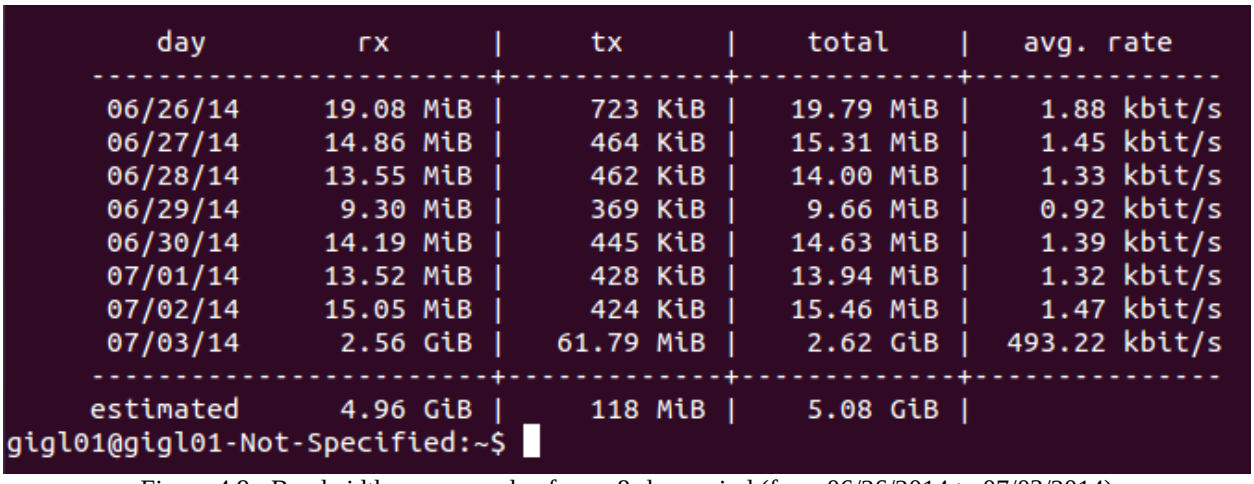


Figure 4.9 : Bandwidth usage per day for an 8-day period (from 06/26/2014 to 07/03/2014)

As you see in Figure 4.9 the busiest day and the least busy day in a sample 8-day period of our web server (from 06/26/2014 to 07/03/2014) are 07/03/2014 and 06/29/2014 respectively.

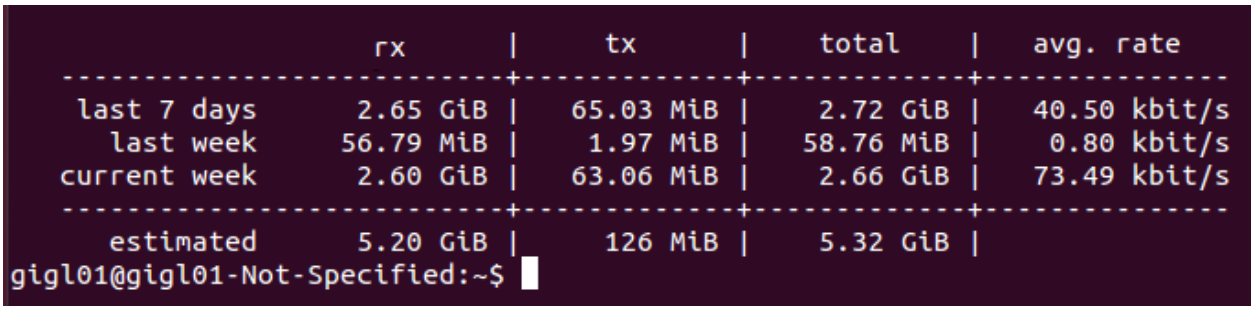


Figure 4.10 : Bandwidth usage per week

The busiest and the least busy weeks are shown in the Figure 4.10 in a sample 2-weeks period of our web server.

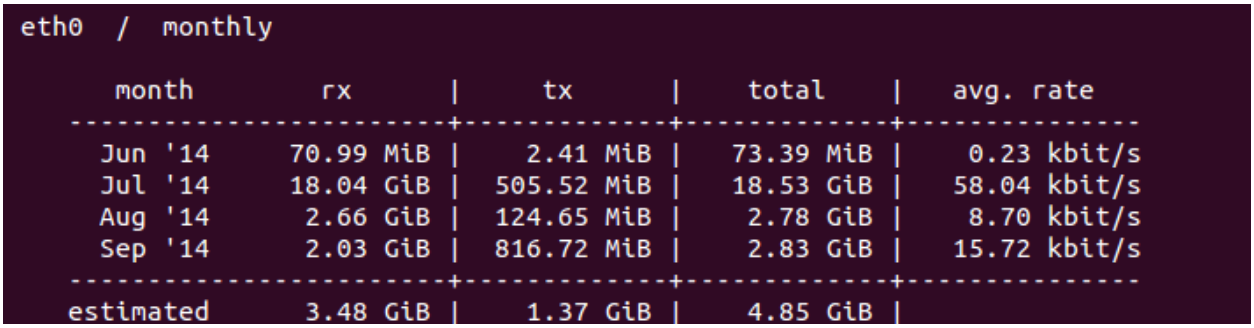


Figure 4.11 : Bandwidth usage per month

Figure 4.11 shows the bandwidth usage in June, July, August and September 2014.

New arrival traffic should be checked with normal situation to detect a change point in one of the above terms (NBWph, NBWpd, NBWpw, NBWpm).

Figures 4.12,4.13, 4.14 and 4.15 present the values of these metrics in different numbers of test and the results of implementing CUSUM for detecting change points in Http attack.

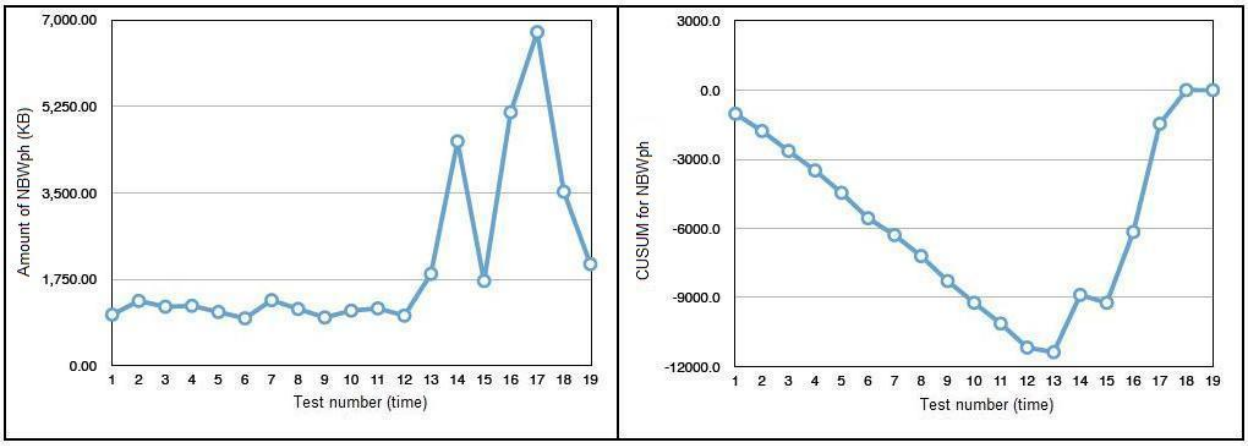


Figure 4.12: variation of NBWph values (left) and CUSUM values in NBWph (right) in Http attack

As it is clear in figure 4.12 (right) a sudden change happened between number of test 12 and 14 in Http attack situation.

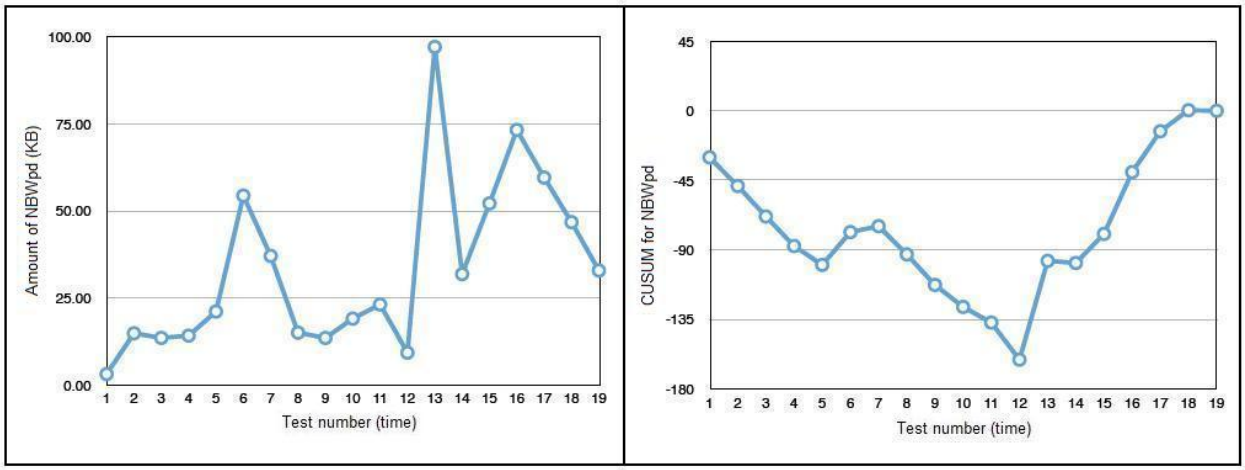


Figure 4.13: variation of NBWpd values (left) and CUSUM values in NBWpd (right) in Http attack

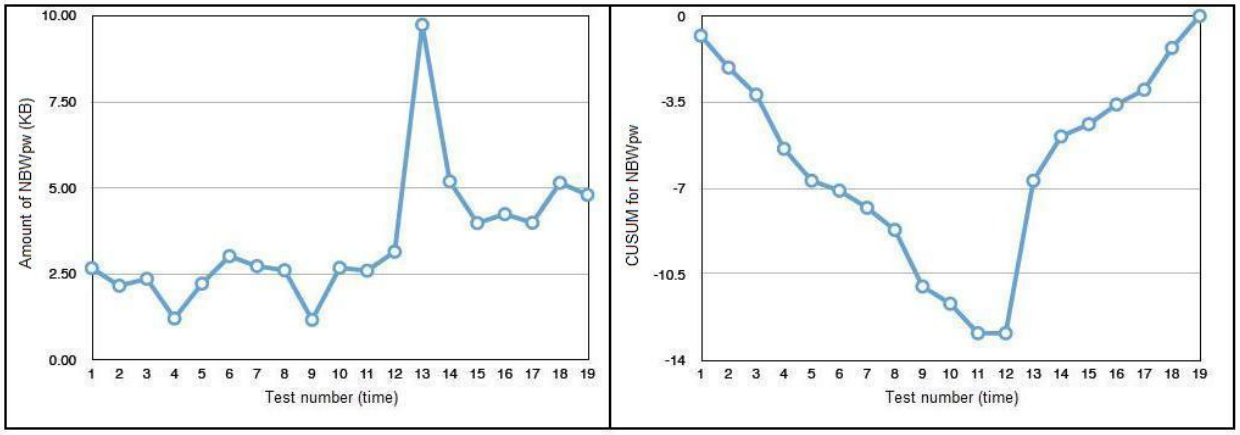


Figure 4.14:variation of NBWpw values (left) and CUSUM values in NBWpw(right) in Http attack

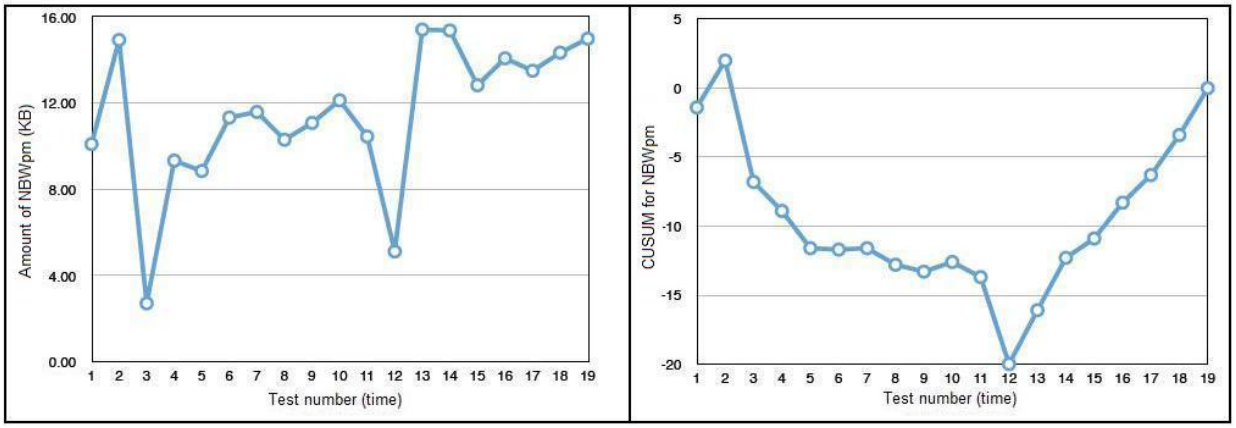


Figure 4.15: variation of NBWpm values(left) and CUSUM values in NBWpm(right) in Http attack

In figure 4.13(right), we can see multiple change points; but as we discussed in chapter 3, modulus of maximum amount of change (max $|S_i|$) will be considered. So in number of test 12 we have the maximum amount of change.

Also it is clear in figure 4.14 (right) that a sudden change happened between number of test 12 and 13 in Http attack situation.

In figure 4.15 (right), modulus of maximum amount of change is in number of test 12.

4.7 Packet information

We used LTTng [87] for sampling and capturing information from traffics. As you can see in Figure 4.16, we could successfully capture the information about flags of many packets in one

sample of the traffic. We analysed these flags and extracted **R(SYN), R(ACK)** and **R(SYN+ACK).**

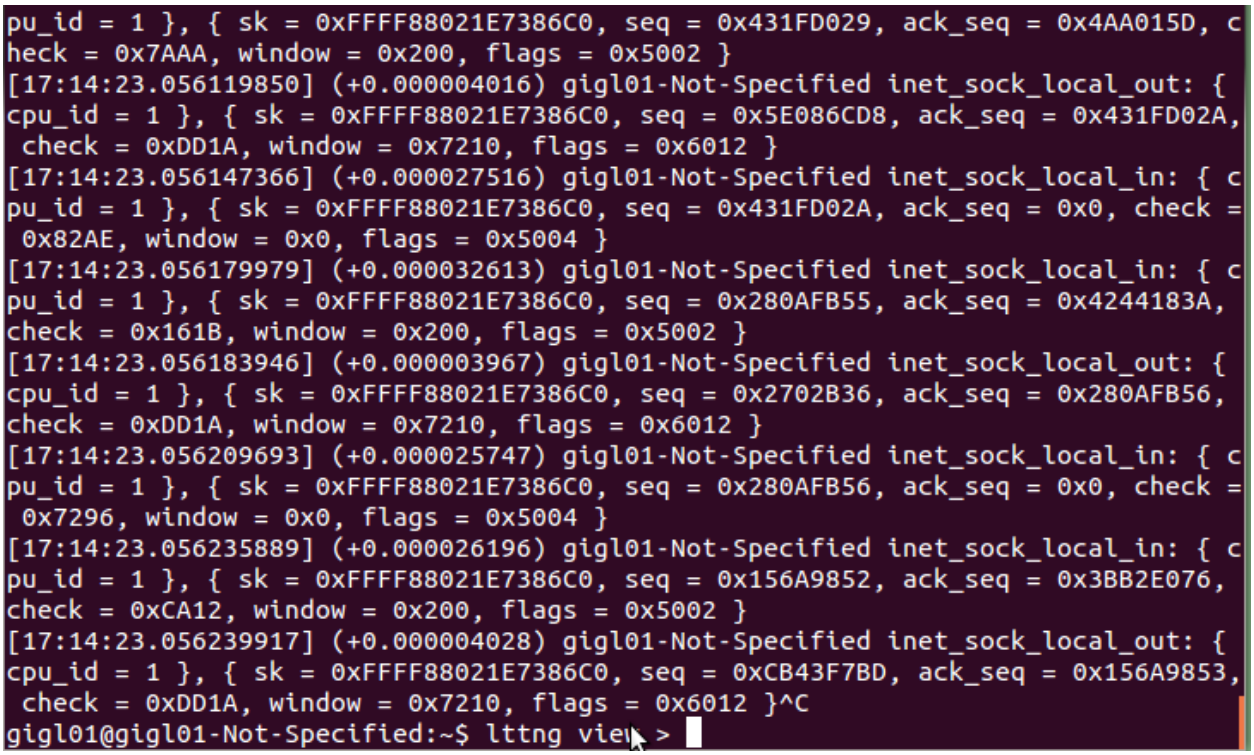


Figure 4.16 : packets information using LTTng

For example as you see in Figure 4.16, the first value of flags is 5002. This number is in hexadecimal format. We first changed it to the binary format and then analyzed it using the TCP header table presented in Figure 4.17. There are 9 bits (from 7 to 15) in the TCP header that indicate the flag information. The most important bits for us are ACK (bit 11) and SYN (bit 14). We extracted these information from packets and calculated **R(SYN), R(ACK)** and **R(SYN+ACK)**. In the above example, the first value of flags indicates that the value of SYN is equal to1 and the value of ACK is equal to 0, so this packet is a SYN packet.

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20	160	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.)																															
...	...	...																															

Figure 4.17 : TCP Header[98]

4.8 Number of packets per second (incoming and outgoing)

Number of packets per second is another metric which is collected by *IPTraf* tool similar to collecting I/O rate as presented in figure 4.3. As we mentioned earlier, Number of packets has been collected in 2 parts, incoming rate and outgoing rate, in webserver virtual machine.

Figures 4.18 and 4.19 show the values of these metrics in different numbers of test and the results of implementing CUSUM that used for detecting change points in TCP SYN Flood attack.

It is completely clear that we have a change point in the number of test 12 in both figures 4.18 and 4.19.

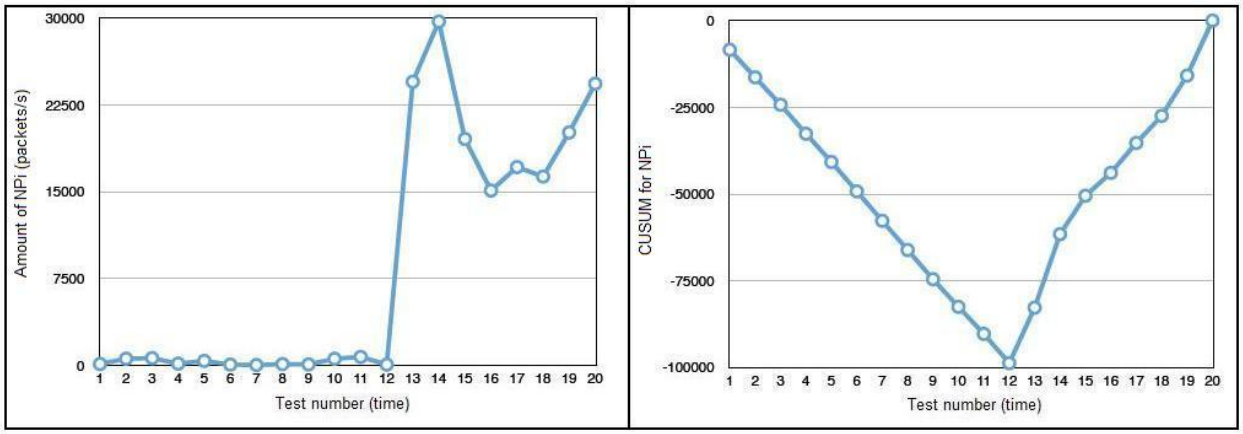


Figure 4.18: variation of NPi values (left) and CUSUM values in NPi(right) in TCP SYN Flood attack

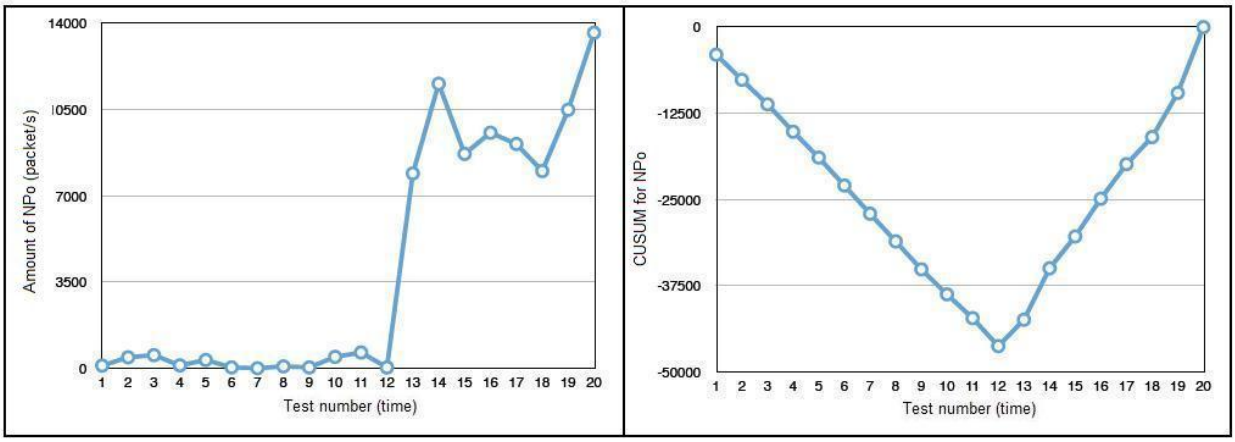


Figure 4.19: variation of NPo values (left) and CUSUM values in NPo(right) in TCP SYN Flood attack

4.9 Number of half-opened connections

The number of half opened connections is measured by using *netstat* command in Linux as you can see in Figure 4.20.

```
gigl01@gigl01-Not-Specified:~$ netstat -an | grep -c SYN_RECV
138
gigl01@gigl01-Not-Specified:~$ netstat -an | grep -c SYN_RECV
67
gigl01@gigl01-Not-Specified:~$ netstat -an | grep -c SYN_RECV
166
```

Figure 4.20 : number of half-opened connection by use of netstat

As expected, and according to our results in different tests by *netstat*, the number of SYN-RECV which is representative of half opened connection in Linux [2] in HTTP attack and database attack are almost the same as the normal situation. But this parameter shows a sudden increase in case of TCP SYN Flood attack.

Figure 4.21(right) indicates this change in TCP SYN Flood attack in number of test 12 clearly.

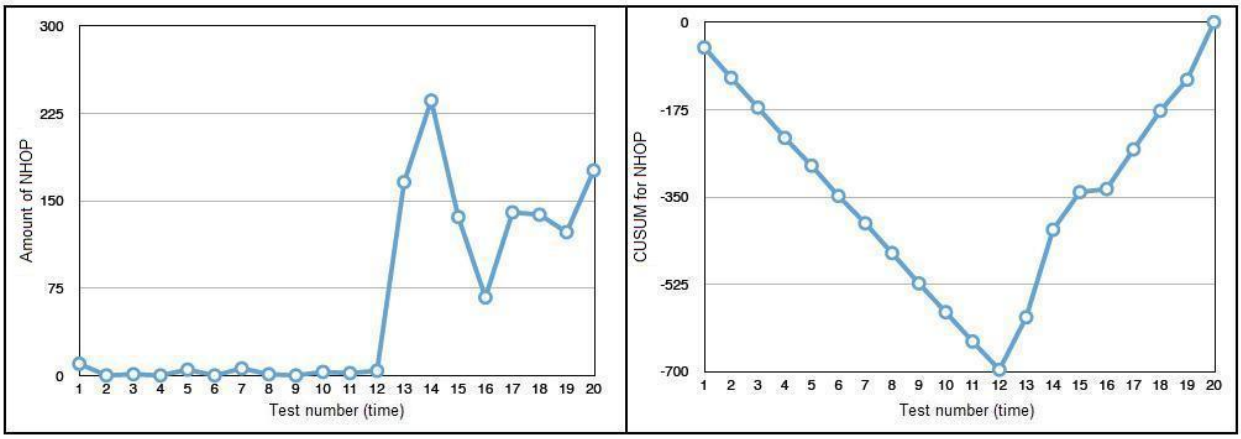


Figure 4.21: variation of NHOP values(left) and CUSUM values in NHOP(right) in TCP SYN Flood attack.

4.10Evaluation and discussion

As discussed earlier, when VMs need more resources (Memory, CPU, Network bandwidth, etc.) they send their requests to hypervisor for increasing the resources and hypervisor allocates the required resources to these VMs; if the hypervisor doesn't have enough resources VM migrates to another hypervisor. We put our framework in the middle of this process so when a request was sent from VMs, this request is first received by our Traffic Statistic Computation & Measurement part. The task of this part is to extract metrics of all VMs which are in connection with the desired VM. Afterwards, the results generated by Traffic Statistic Computation & Measurement part (TSP, IOWi,…), will be sent to Detecting Unit where our proposed detecting algorithm can detect what the situation of new arrival traffic is and can calculate percentage of each type of attack as well (Figure 3.1).

Based on the result of this detection, if one of the three attacks is detected with a high percentage, allocating the resources will be limited otherwise resources will be allocated to VM. The critical value of the attack percentage to make a decision depends on the sensitivity of the system. For more sensitive system, even attacks with low percentage causes limiting resources.

In this chapter we evaluate our system using 2 criteria: Metric and accuracy; that we are going to explain in the following sections:

4.10.1 Accuracy evaluation

In order to evaluate the accuracy of our algorithm, we tested our detecting model and compared it with previous works. For this purpose, we choose Snort[99]. Snort is a free and open source IDS (intrusion detection system) which can perform real-time traffic analysis and detect the attacks [100]. The result showed that our model work with higher rate of correct detection compared to previous IDS like Snort. Figure 4.22 shows the result of comparison between the rate of correct detection in our model and in snort IDS.

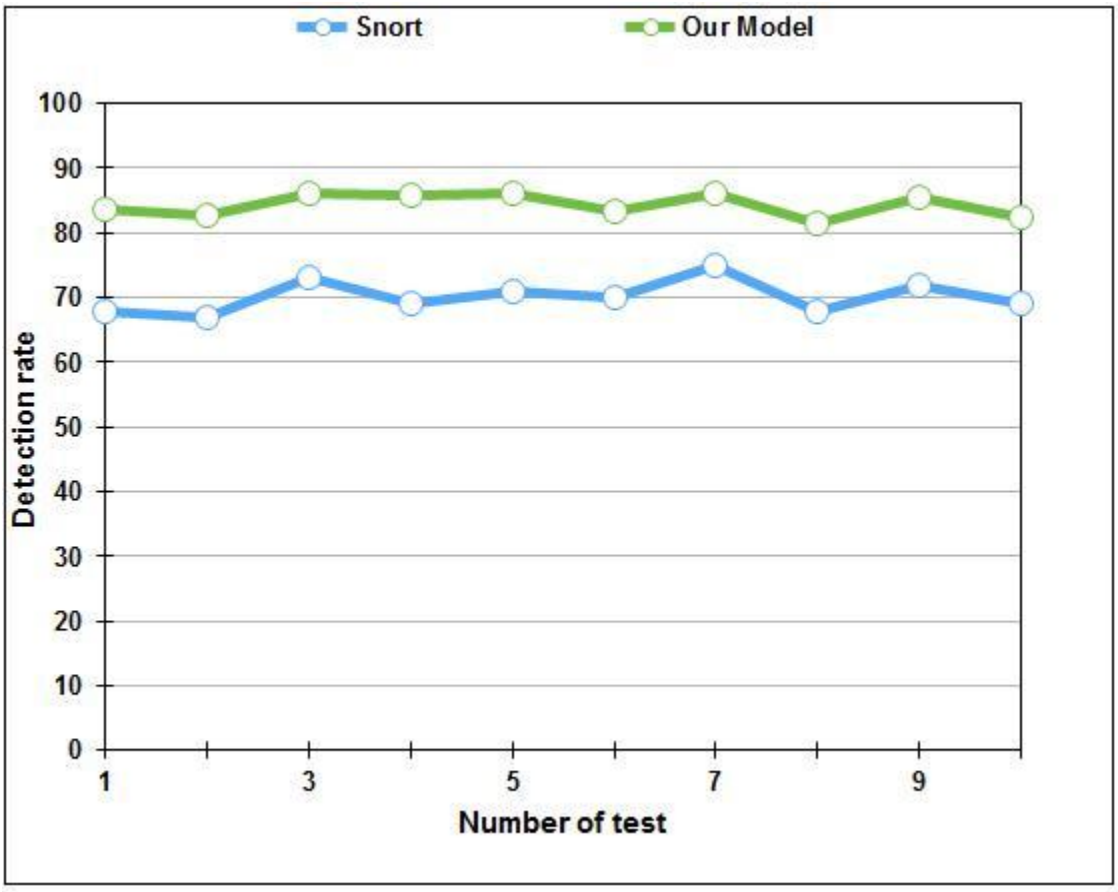


Figure 4.22: rate of correct detection comparison

As it is clear in figure 4.22, the rate of correct detection in our model in average is about 85 % approximately. However, the average of rate of correct detection in snort is 70% approximately.

In view of the above result we can conclude that the accuracy of our detection model is reliable and very well acceptable.

4.10.2 Metrics evaluation

As mentioned earlier, we used all the metrics of different types of attack for detecting each type of attack in our framework. It means that the metrics of one kind of attack play an important role to detect another type and vice versa. In order to evaluate the performance of our detection model, we compared our framework efficiency with algorithms that use only the special metrics of one attack to detect that attack without considering the metrics of other attacks. We named these three algorithms A1, A2 and A3 as follow:

- A1: Algorithms which detect HTTP attacks based on specific metrics of HTTP attacks.
- A2: Algorithms which detect TCP SYN flood attacks based on specific metrics of TCP SYN flood attacks.
- A3: Algorithms which detect Database attacks based on specific metrics of Database attacks.

In order to evaluate the efficiency of our metrics we need to test and compare them with A1, A2 and A3 without performing our detection model. One option is using a machine learning algorithm for evaluating our metrics. We are going to see how much the percentage of correctly classify traffic is when a machine learning algorithm uses our metrics and compare it with the cases that the machine learning algorithm uses A1, A2 or A3 metrics.

For this purpose, we have selected a popular machine learning algorithm called Neural Network. This algorithm has been widely used in previous works in security concept specially in DDOS attack in both cloud computing[101, 102] and old network (non- virtualization systems)[103].

WEKA tool has been used to implement Neural Network. WEKA is a machine learning workbench which helps to use different machine learning algorithms for different real-word problems [104]. *MultilayerPerceptron* is a function in WEKA which represents Neural Network. We have configured Neural Network and trained it by a set of samples (65% of our data) which all include the previously explained 18 metrics (TSP,IODi,…,NHOP). Afterward, we have tested the algorithm using the rest of samples (35% of our data) which all include 18 metrics as well.

As a result, Neural Network could precisely classify traffic where 97.06% of traffic has been correctly classified and only 2.94 % has been incorrectly classified. Figure 4.23 demonstrates the result of our testing by Neural Network.

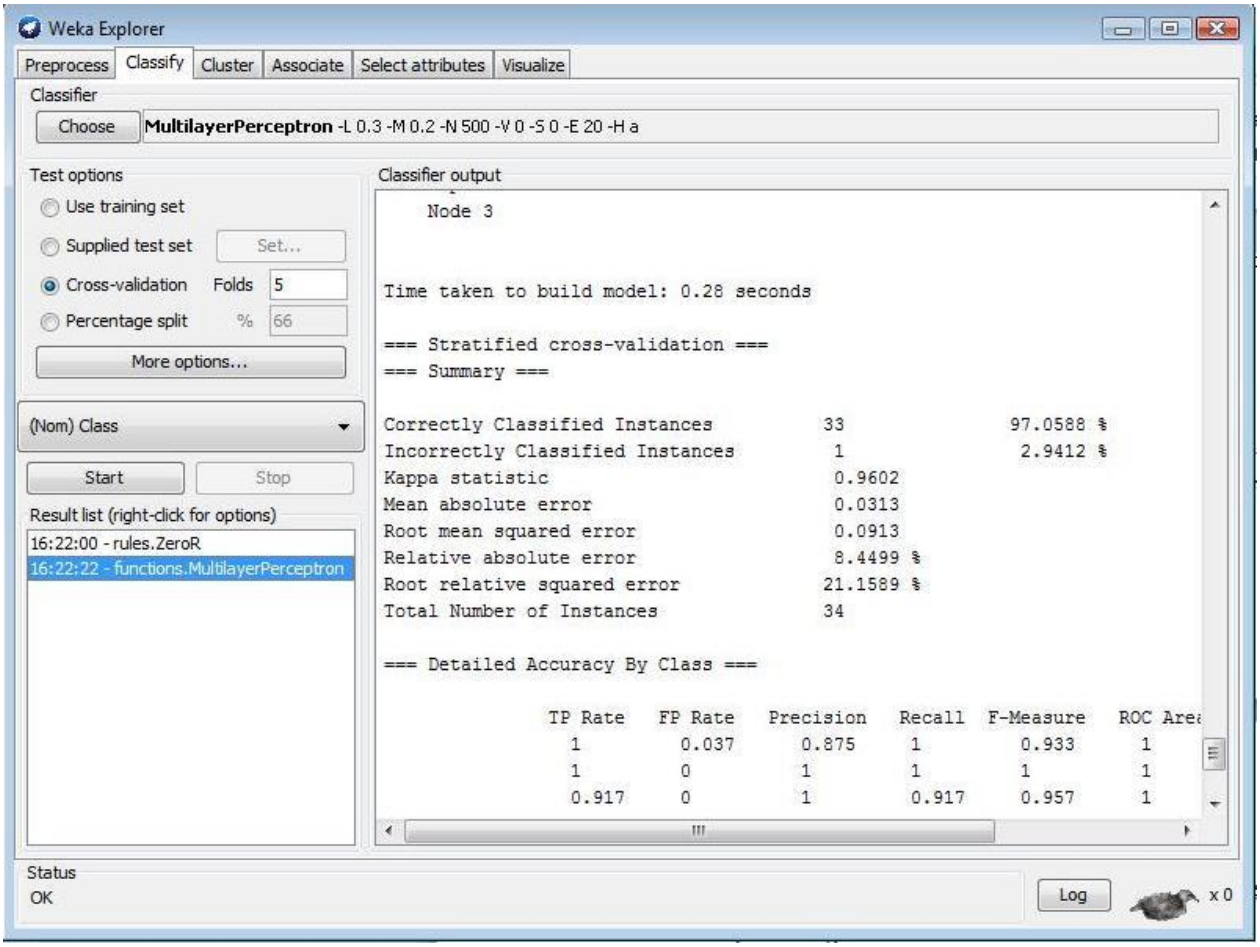


Figure 4.23: Testing our framework metrics by Neural Network

We perform neural network configuration again based on A1 metrics, A2 metrics and A3 metrics separately. Then we compared neural network classification based on our metrics and based on A1, A2 and A3 metrics. Figure 4.24 shows the result of this comparison.

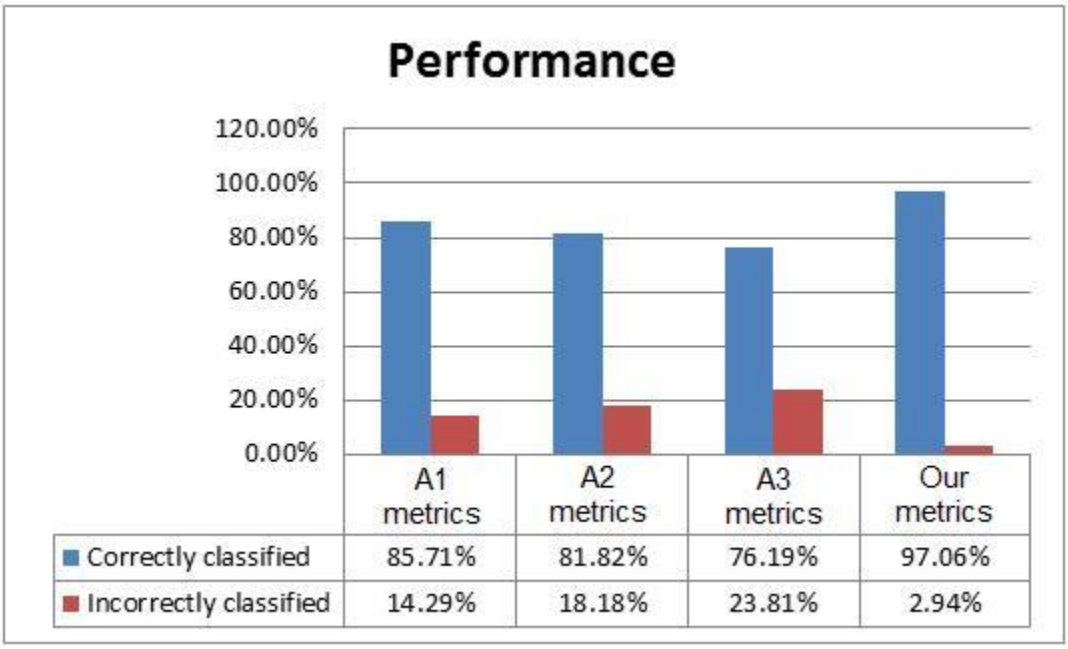


Figure 4.24 : Evaluating our metrics by using Neural network

As it's shown in Figure 4.24 our metrics have a tremendously higher percentage of correctly classified attacks and a much lower percentage of incorrectly classified attack. This test by Neural Network with 97.06 % correct classification rate can prove that we have chosen proper and effective metrics.

Furthermore, our first and obvious advantage of our model is that we are able to detect every 3 types of attacks in our framework while other algorithms can detect only one type of attack.

CHAPTER 5 CONCLUSION

In the frame of this thesis, we discussed about security which is a big concern in cloud computing. We investigated DDoS attacks and a specific type of these attacks which is called TCP SYN flooding attacks. These are common and popular attacks in cloud computing. EDoS which is a new and exceptional type of DDoS attack can also influence Cloud through a DDoS and can make economic problem for cloud costumer and provider. We investigated Http attack and Database attack as special types of EDoS and DDoS attacks. In order to successfully confront the attack and to be able to prevent the system against it the most important step is to design a reliable and powerful framework for attack detection.

As it was shown in the literature review, there are a lot of algorithms in traditional (non-virtualization) system for detecting DDoS attacks which work well in non-virtualization system. However, these algorithms are useless in cloud environment if we don’t apply the required modification.

On the other hand, the majority of existing algorithms for detecting DDoS attacks in cloud computing environment work based on packet information. Since the packet in DDoS attack is identical to packet in the normal situation, these algorithms are not reliable enough for DDoS detection.

Also, the current algorithms for EDoS attacks are useful only for one attack and to our best knowledge there is not any global model to detect all types of attacks. Moreover, they have mainly focused only on finding a solution for the defense and mitigation of EDoS attacks in cloud computing.

In this work we proposed a novel and inclusive model for detecting different types of EDoS and DDoS attacks. We used HTTP attack as a good representative of the Bandwidth – Consuming EDoS attacks. To address EDoS as an "Attack that targets specific applications" we considered Database as the targeted application. Moreover, as a delegate for connection–layer exhaustion attacks, we nominated TCP SYN Flood attack. At the next step, we collected features which are related to traffic and resource usage in each attack. These features formed the metrics of our detection model. In designing our model, we used the metrics related to all 3 types of attacks and consequently we could successfully introduce a global and inclusive algorithm which worked perfectly for all three of HTTP attack, Database attack and TCP SYN Flood attack no matter

what time each attack happened. Therefore, instead of using three different algorithms for detecting these attacks, we are able to use a generalized algorithm.

The accuracy of our algorithm was then investigated by comparing its performance with Snort. Achieving a higher rate of correct detection in our model proved the high accuracy of the designed algorithm.

In order to evaluate the efficiency of our model's metrics we compared them with the metrics of other algorithms which considered only one type of attack; however, this needed to be done without performing our detection model. For this purpose, we used Neural Network that has been widely used in previous works in security concept in cloud computing. The result showed that our metrics have a tremendously higher percentage of correctly classified attacks and a much lower percentage of incorrectly classified attacks. Thus using this global algorithm is much more efficient in detecting the attacks compared to using three separate algorithms since metrics of one kind of attack plays an important role to detect another type.

Furthermore, this precise detection can be considered as a defense itself because the services will be provided only to VMs that are in Normal situation; therefore we prevent the migration of attacks from VM to the hypervisor. This is a temporary defense for the system till finding a permanent solution in the future.

Although our algorithm is highly reliable and extremely accurate in detecting all three mentioned attacks, it has a limitation. If the Database attack and Http attack occur simultaneously in the system, the traffic and resource usage are very similar to heavy and busy traffic in the normal situation. Therefore the accuracy of detection will degrade in this case. However, the possibility of having these two attacks concurrently in the system is very low and almost negligible and that's why we ignored this situation in our assumptions.

This work can be further expanded to find features for other attacks in addition to HTTP attack, Database attack and TCP SYN Flood. A very interesting elaboration to this work would be to allocate weights to the metrics and then to design a machine learning algorithm that can automatically predict and detect other types of attacks using this expanded model.

REFERENCES

[1] L. McMillan, E. White, M. Terpstra, P. Romanski, and N. Ivanov, "Twenty-One Experts Define Cloud Computing," *Virtualization Journal,* 2009.

[2] M. S. Bogdanoski, T. Risteski, A., "Analysis of the SYN Flood DoS Attack," *Computer Network and Information Security* 2013.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing " National Institute of Standards and Technology2011.

[4] B. Furht and A. Escalante, *Handbook of Cloud Computing*: Springer, 2010.

[5] S. VivinSandar and S. Shenai, "Economic Denial of Sustainability (EDoS) in Cloud Services using HTTP and XML based DDoS Attacks," *International Journal of Computer Applications,* 2012.

[6] NSFOCUS, "Bandwidth Consumption DDoS Attacks and Mitigation Methods," NSFOCUS2013.

[7] (2010). *Application Denial of Service*. Available: https://www.owasp.org/index.php/Application_Denial_of_Service

[8] E. Chai, "Incapsula's Five-Ring Approach to Application layer DDoS Protection," in *Incapsula*ed, 2013.

[9] T. Hsin-Yi, M. Siebenhaar, A. Miede, H. Yu-Lun, and R. Steinmetz, "Threat as a Service?: Virtualization's Impact on Cloud Security," *IT Professional,* vol. 14, pp. 32-37, 2012.

[10] (29 Dec 2014). *http://www.salesforce.com/uk/socialsuccess/cloud-computing/the-complete-history-of-cloud-computing.jsp*.

[11] R. Buyya, C. Yeo, S., and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," *CoRR Journal,* 2008.

[12] (2010). *Definition of Cloud Computing, incorporating NIST and G-Cloud views*. Available: https://www.katescomment.com/definition-of-cloud-computing-nist-g-cloud/comment-page-1/#comment-1996.

[13] R. L. Grossman, "The Case for Cloud Computing," *IT Professional,* vol. 11, pp. 23-27, 2009.

[14] F. Baiardi and D. Sgandurra, "Securing a Community Cloud," in *Distributed Computing Systems Workshops (ICDCSW), 2010 IEEE 30th International Conference on*, 2010, pp. 32-41.

[15] S. Tanimoto, Y. Sakurada, Y. Seki, M. Iwashita, S. Matsui, H. Sato*, et al.*, "A Study of Data Management in Hybrid Cloud Configuration," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2013 14th ACIS International Conference on*, 2013, pp. 381-386.

[16] A. Goyal and S. Dadizadeh, "A Survey on Cloud Computing," Technical Report for CS 508, 2009.

[17] "http://aws.amazon.com/ec2/," Jan 2015.

[18] "http://azure.microsoft.com/en-us/," 2014.

[19] M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono, "On Technical Security Issues in Cloud Computing," in *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, 2009, pp. 109-116.

[20] W. Chuliang, G. Minyi, L. Yuan, and L. Minglu, "Hybrid CPU Management for Adapting to the Diversity of Virtual Machines," *Computers, IEEE Transactions on,* vol. 62, pp. 1332-1344, 2013.

[21] K. Caraher and M. Nott, "CDW Server Virtualization Life Cycle Report," 2010.

[22] D. P. Botero, J. Szefer, and R. B. Lee, "Characterizing hypervisor vulnerabilities in cloud computing servers," presented at the Security in cloud computing,ACM, Hangzhou, China, 2013.

[23] K. J. Higgins, "Vm's create potential risks," Technical report, darkREADING,2007.

[24] L. Yunfa, L. Wanqing, and J. Congfeng, "A Survey of Virtual Machine System: Current Technology and Future Trends," in *Electronic Commerce and Security (ISECS), 2010 Third International Symposium on*, 2010, pp. 332-336.

[25] A. Mann, "The pros and cons of virtualization," BTQ2007.

[26] J. Kirch, "Virtual machine security guidelines," The center for Internet Security2007.

[27] J. Pfoh, C. Schneider, and C. Eckert, "Exploiting the x86 Architecture to Derive Virtual Machine State Information," in *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, 2010, pp. 166-175.

[28] J. McTigue, "Virtualization Management Survey," InformationWeek Reports2013

[29] X. Padala, Z. Z. Wang, S. Singhal, and K. Shin, "Performance evaluation of virtualization technologies for server consolidation," HP Labs Tec.2007.

[30] J. N. Matthews, E. Dow, T. Deshane, W. Hu, J. Bongio, P. F. Wilbur*, et al.*, *Running Xen: A Hands-On Guide to the Art of Virtualization*, 2008.

[31] B. Casselman, S. Kaplan, and T. Reeser, *Citrix XenApp Platinum Edition for Windows: The Official Guide.* , 2008.

[32] (Jan 2015). *http://www.openstack.org/*.

[33] H. Tianfield, "Security issues in cloud computing," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, 2012, pp. 1082-1089.

[34] X. Zhifeng and X. Yang, "Security and Privacy in Cloud Computing," *Communications Surveys & Tutorials, IEEE,* vol. 15, pp. 843-859, 2013.

[35] Y. Kejiang, J. Xiaohong, H. Dawei, C. Jianhai, and W. Bei, "Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 267-274.

[36] D. Versick and D. Tavangarian, "Reducing Energy Consumption by Load Aggregation with an Optimized Dynamic Live Migration of Virtual Machines," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, 2010, pp. 164-170.

[37] S. Sahni and V. Varma, "A Hybrid Approach to Live Migration of Virtual Machines," in *Cloud Computing in Emerging Markets (CCEM), 2012 IEEE International Conference on*, 2012, pp. 1-5.

[38] G. Xu, J. Pang, and X. Fu, "A load balancing model based on cloud partitioning for the public cloud," *Tsinghua Science and Technology,* vol. 18, pp. 34-39, 2013.

[39] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, N. Nitin*, et al.*, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization," in *Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on*, 2012, pp. 3-8.

[40] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski*, et al.*, "Above the Clouds: A Berkeley View of Cloud Computing," Electrical Engineering and Computer Sciences University of California at Berkeley UCB/EECS-2009-28, 2009.

[41] Z. Minqi, Z. Rong, X. Wei, Q. Weining, and Z. Aoying, "Security and Privacy in Cloud Computing: A Survey," in *Semantics Knowledge and Grid (SKG), 2010 Sixth International Conference on*, 2010, pp. 105-112.

[42] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "RAID: high-performance, reliable secondary storage," *ACM Comput. Surv.,* 1994.

[43] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system " presented at the 19th Symposium on Operating Systems Principles, 2003.

[44] I. M. Khalil, A. Khreishah, S. Bouktif, and A. Ahmad, "Security Concerns in Cloud Computing," in *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*, 2013, pp. 411-416.

[45] R. Kui, W. Cong, and W. Qian, "Security Challenges for the Public Cloud," *Internet Computing, IEEE,* vol. 16, pp. 69-73, 2012.

[46] Z. Birnbaum, L. Bingwei, A. Dolgikh, C. Yu, and V. Skormin, "Cloud Security Auditing Based on Behavioral Modeling," in *Services (SERVICES), 2013 IEEE Ninth World Congress on*, 2013, pp. 268-273.

[47] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the hypervisor attack surface for a more secure cloud," presented at the Proceedings of the 18th ACM conference on Computer and communications security, Chicago, Illinois, USA, 2011.

[48] (Jan 2015). *Rootkit, http://en.wikipedia.org/wiki/Rootkit*.

[49] M. Bamiah, S. Brohi, S. Chuprat, and M. N. Brohi, "Cloud implementation security challenges," in *Cloud Computing Technologies, Applications and Management (ICCCTAM), 2012 International Conference on*, 2012, pp. 174-178.

[50] R. Yeluri, E. Castro-Leon, R. R. Harmon, and J. Greene, "Building Trust and Compliance in the Cloud for Services," in *SRII Global Conference (SRII), 2012 Annual*, 2012, pp. 379-390.

[51] J. S. Reuben, "A Survey on Virtual Machine Security," TKK T-110.5290 Seminar on Network Security2007.

[52] U. Oktay, M. A. Aydin, and O. K. Sahingoz, "Circular chain VM protection in AdjointVM," in *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on*, 2013, pp. 93-97.

[53] S. U. Muthunagai, C. D. Karthic, and S. Sujatha, "Efficient access of Cloud Resources through virtualization techniques," in *Recent Trends In Information Technology (ICRTIT), 2012 International Conference on*, 2012, pp. 174-178.

[54] R. Shea and L. Jiangchuan, "Understanding the impact of Denial of Service attacks on Virtual Machines," in *Quality of Service (IWQoS), 2012 IEEE 20th International Workshop on*, 2012, pp. 1-9.

[55] Y. G. Dantas, V. Nigam, and I. E. Fonseca, "A Selective Defense for Application Layer DDoS Attacks," in *Intelligence and Security Informatics Conference (JISIC), 2014 IEEE Joint*, 2014, pp. 75-82.

[56] Q. Liao, H. Li, S. Kang, and C. Liu, "Feature extraction and construction of application layer DDoS attack based on user behavior," in *Control Conference (CCC), 2014 33rd Chinese*, 2014, pp. 5492-5497.

[57] J. Nicholls. (2012) The changing face and growing threat of DDoS. *SC Magazine*.

[58] A. Chonka, Z. Wanlei, and X. Yang, "Protecting web services with Service Oriented Traceback Architecture," in *Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on*, 2008, pp. 706-711.

[59] A. Bouayad, A. Blilat, N. El Houda Mejhed, and M. El Ghazi, "Cloud computing: Security challenges," in *Information Science and Technology (CIST), 2012 Colloquium in*, 2012, pp. 26-31.

[60] S. Padmanabhuni, V. Singh, K. M. Senthil Kumar, and A. Chatterjee, "Preventing Service Oriented Denial of Service (PreSODoS): A Proposed Approach," in *Web Services, 2006. ICWS '06. International Conference on*, 2006, pp. 577-584.

[61] J. Stewart. (2007, Jan 2015). *HTTP DDoS Attack Mitigation Using Tarpitting*. Available: http://www.secureworks.com/cyber-threat-intelligence/threats/ddos/

[62] incapsula. (March 2015). *Distributed Denial of Service Attack (DDoS) Definition*. Available: http://www.incapsula.com/ddos/ddos-attacks/

[63] Y. Lanjuan, Z. Tao, S. Jinyu, W. JinShuang, and C. Ping, "Defense of DDoS attack for cloud computing," in *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, 2012, pp. 626-629.

[64] A. Chonka, Xiang, Y., Zhou,W., Bonti,A., "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks," *Journal of Network and Computer Applications,* 2011.

[65] W. Jin, Y. Xiaolong, and L. Keping, "A new relative entropy based app-DDoS detection method," in *Computers and Communications (ISCC), 2010 IEEE Symposium on*, 2010, pp. 966-968.

[66] A. Chonka, Z. Wanlei, and X. Yang, "Defending Grid Web Services from XDoS attacks by SOTA," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, 2009, pp. 1-6.

[67] L. Ang, G. Lin, and X. Kuai, "Fast Anomaly Detection for Large Data Centers," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1-6.

[68] A. Anand, M. Dhingra, J. Lakshmi, and S. K. Nandy, "Resource usage monitoring for KVM based virtual machines," in *Advanced Computing and Communications (ADCOM), 2012 18th Annual International Conference on*, 2012, pp. 66-70.

[69] K. Cho, R. Kaizaki, and O. Nakamura, "Detection Denial of Service Attacks Using AGURI," presented at the International Conference Telecommunications, Beijing China, 2002.

[70] A. G. Tartakovsky, B. L. Rozovskii, R. B. Blazek, and K. Hongjoong, "A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods," *Signal Processing, IEEE Transactions on,* vol. 54, pp. 3372-3382, 2006.

[71] M. T. Gil and M. Poletto, "MULTOPS: a data-structure for bandwidth attack detection," presented at the In Proceedings of 10th Usenix Security Symposium, 2001.

[72] S. Noh, C. Lee, K. Choi, and G. Jung, "Detecting Distributed Denial of Service (DDoS) Attacks through Inductive Learning," in *Intelligent Data Engineering and Automated Learning*. vol. 2690, J. Liu, Y.-m. Cheung, and H. Yin, Eds., ed: Springer Berlin Heidelberg, 2003, pp. 286-295.

[73] R. Jalili, F. Imani-Mehr, M. Amini, and H. Shahriari, "Detection of Distributed Denial of Service Attacks Using Statistical Pre-processor and Unsupervised Neural Networks," in *Information Security Practice and Experience*. vol. 3439, R. Deng, F. Bao, H. Pang, and J. Zhou, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 192-203.

[74] A. B. Kulkarni, S. F. Bush, and S. C. Evans, "Detecting Distributed Denial-of-Service Attacks Using Kolmogorov Complexity Metrics," GE Research & Development CenterFebruary 2002.

[75] M. H. Sqalli, F. Al-Haidari, and K. Salah, "EDoS-Shield - A Two-Steps Mitigation Technique against EDoS Attacks in Cloud Computing," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, 2011, pp. 49-56.

[76] M. Naresh Kumar, P. Sujatha, V. Kalva, R. Nagori, A. K. Katukojwala, and M. Kumar, "Mitigating Economic Denial of Sustainability (EDoS) in Cloud Computing Using In-cloud Scrubber Service," in *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on*, 2012, pp. 535-539.

[77] W. Alosaimi and K. Al-Begain, "An Enhanced Economical Denial of Sustainability Mitigation System for the Cloud," in *Next Generation Mobile Apps, Services and Technologies (NGMAST), 2013 Seventh International Conference on*, 2013, pp. 19-25.

[78] M. Masood, Z. Anwar, S. A. Raza, and M. A. Hur, "EDoS Armor: A cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments," in *Multi Topic Conference (INMIC), 2013 16th International*, 2013, pp. 37-42.

[79] Z. A. Baig and F. Binbeshr, "Controlled Virtual Resource Access to Mitigate Economic Denial of Sustainability (EDoS) Attacks against Cloud Infrastructures," in *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, 2013, pp. 346-353.

[80] F. Al-Haidari, M. H. Sqalli, and K. Salah, "Enhanced EDoS-Shield for Mitigating EDoS Attacks Originating from Spoofed IP Addresses," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, 2012, pp. 1167-1174.

[81] A. Koduru, T. Neelakantam, and S. M. Saira Bhanu, "Detection of Economic Denial of Sustainability Using Time Spent on a Web Page in Cloud," in *Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on*, 2013, pp. 1-4.

[82] R. Shea and L. Jiangchuan, "Performance of Virtual Machines Under Networked Denial of Service Attacks: Experiments and Analysis," *Systems Journal, IEEE,* vol. 7, pp. 335-345, 2013.

[83] V. Nigam, S. Jain, and K. Burse, "Profile Based Scheme against DDoS Attack in WSN," in *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, 2014, pp. 112-116.

[84] V. Venkata Ramana, P. Shilpa Choudary, and M. B. Dhone, "Analysis & Study of Application Layer Distributed Denial of Service Attacks for Popular Websites," *International Journal of Computer Science and Telecommunications,* vol. 2, 2011.

[85] Y. Chengxu and Z. Kesong, "Detection of application layer distributed denial of service," in *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, 2011, pp. 310-314.

[86] K. Geetha and N. Sreenath, "SYN flooding attack — Identification and analysis," in *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*, 2014, pp. 1-7.

[87] M. Desnoyers, Dagenais, M., "LTTng: Tracing across execution layers,from the hypervisor to user-space. ," presented at the Proceedings of the Ottawa Linux Symposium, 2008.

[88] V. Montes De Oca, D. R. Jeske, Q. Zhang, C. Rendon, and M. Marvasti, "A cusum change-point detection algorithm for non-stationary sequences with application to data network surveillance," *Journal of Systems and Software,* vol. 83, pp. 1288-1297, 7// 2010.

[89] A. Taylor. (2014). *Change-Point Analysis: A Powerful New Tool For Detecting Changes*. Available: http://www.variation.com/cpa/tech/changepoint.html

[90] (17 july 2014). *http://wiki.qemu.org/Main_Page*.

[91] (2015). *httpd - Apache Hypertext Transfer Protocol Server*. Available: http://httpd.apache.org/docs/2.2/programs/httpd.html

[92] (2015). *MySQL*. Available: https://www.mysql.com/

[93] (2015). *Netsniff-ng toolkit*. Available: http://netsniff-ng.org/

[94] *hping*. Available: http://www.hping.org/

[95] (April 21, 2015). *Denial-of-service Attack – DoS using hping3 with spoofed IP in Kali Linux*. Available: http://www.blackmoreops.com/2015/04/21/denial-of-service-attack-dos-using-hping3-with-spoofed-ip-in-kali-linux/

[96] (May 2013). *httpflooder,DoS/DDoS over HTTP Tool*. Available: https://code.google.com/p/httpflooder/wiki/Usage

[97] *LOADRUNNER*. Available: http://www8.hp.com/us/en/software-solutions/loadrunner-load-testing/index.html

[98] (23 july 2014). *http://en.wikipedia.org/wiki/Transmission_Control_Protocol*.

[99] J. Choi, C. Choi, B. Ko, and P. Kim, "A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment," *Soft Comput.,* vol. 18, pp. 1697-1703, 2014.

[100] "Snort (software)," in *Wikipedia*, ed, 2015.

[101] S. Sabnani, V., "Computer Security: A Machine Learning Approach," ed. Royal Holloway, University of London, 07 January 2008.

[102] B. Joshi, A. S. Vijayan, and B. K. Joshi, "Securing cloud computing environment against DDoS attacks," in *Computer Communication and Informatics (ICCCI), 2012 International Conference on*, 2012, pp. 1-5.

[103] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE,* vol. 10, pp. 56-76, 2008.

[104] G. Holmes, A. Donkin, and I. H. Witten, "WEKA: a machine learning workbench," in *Intelligent Information Systems,1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, 1994, pp. 357-361.